PERFECTION IN AUTOMATION
www.br-automation.com

TM223

Automation Studio 진단툴

버전 정보

버전	날짜	수정내역	번역	검수
1.0	2016.11.11	첫번째 버전 TM223TRE.40-ENG_Automation Studio Diagnostics_V4200	정진강	임은

Table 1: Versions

선행 및 필요 조건

트레이닝 모듈	TM210 – Automation Studio 사용하기 TM213 – Automation Runtime
소프트웨어	Automation Studio 4.2
하드웨어	X20CP1586

목차

1 소개

Automation Studio 그리고 Automation Runtime 은 기계의 **프로그래밍**과 **시운전**, 런타임 시스템 **분석**, 그리고 **서비스 동작** 등에 대한 몇가지 진단 기능을 제공한다.

진단 과정의 시작은 어플리케이션이나 그 상황에 맞는 툴을 선택하는 것이다.

그림 1: Diagnostics (진단)

진단 기능은 사용자가 Automation Studio 가 있을 때나 없을 때 기록되는 데이터를 분석하기 위해 특별히 Automation Runtime 에 포함되어 있다.

System Diagnostics Manager(SDM)는 Automation Runtime V3.0 과 함께 지원하는 통합 컴포넌트이다.

1.1 학습목표

이 트레이닝 모듈은 다양한 진단 툴 사용법을 안내하기 위해서 다양한 진단 가능 상황(프로그래밍, 시운전 그리고 서비스)을 예를 들어 설명한다.

- 올바른 진단 툴을 선택하는 기준에 대해 배울 것이다.
- 일반적인 시스템 정보를 평가하고 저장하는 방법을 배울 것이다.
- process 값을 관찰하고 기록하는 방법을 배울 것이다.
- 시스템과 어플리케이션을 진단하기 위한 가능성에 대해 배울 것이다.
- Automation Runtime 설정 옵션이 어떤 진단 툴과 관련이 있는지 배울 것이다.

2 정확한 진단툴

정확한 진단툴을 선택해야 빠르고 효율적으로 문제의 원인을 알아낼 수 있다.

스스로 무관한 데이터를 분석하거나 해결책을 찾기 위해 다른이에게 데이터를 보내고 결정을 맡기는 것은 상당한 시간이 걸릴수 있다.

Logger 를 사용하면 사이클 타임 위반(cycle time violation)을 알 수 있다. 그러나, 이런 경우에 Logger 를 사용하여 사이클 타임 위반을 진단하는 것은 최선책이 아니다.

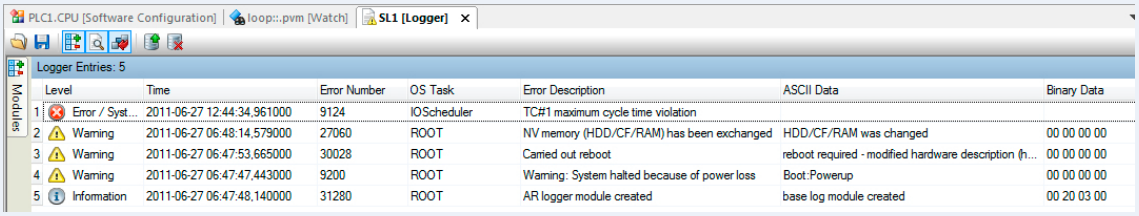

그림 2: Logger 창에서 사이클 타임 위반(cycle time violation)

Logger 에서 에러의 원인은 task class #1 에서 발생한 사이클 타임위반이다. 또한 백트레이스 데이터(backtrace data)는 사이클 타임 위반이 발생한 테스크(task)를 알려준다.

상황 1
높은 우선순위를 가지는 테스크(task)에 의해 인터럽트된 하나의 테스크가 있는 멀티 태스킹 시스템을 보면, 높은 우선순위의 테스크가 cycle time violation 의 원인이다. 이는 높은 우선순위의 테스크가 이 사이클에서 오랜 실행시간 작동하기 때문이며, 이것은 Logger 에서 사용되는 테스크가 더이상 설정된 사이클 타임과 허용 용량안에서 작동하지 않는다는 것을 의미한다.

상황 2
몇몇 테스크들은 같은 테스크 클래스 안에서 다른 테스크가 실행될 때마다 주기적으로 한번 실행된다. 만약 이런 테스크들 중 하나가 오랜 시간이 걸린다면, Logger 에서 보여지는 테스크는 cycle time violation 의 원인이 되지 않을 것이다.

두 경우 모두, 에러를 결정하는데 있어서 Logger 는 **잘못된 진단 툴**이 될 수 있다. 이 문제는 오직 Profiler (5.1 “Profiler 설정과 데이터 평가”) 를 사용해서 검출할 수 있으며, 이는 개별 테스크의 시간 순 시퀀스와 실행되는데 필요한 시간을 보여준다.

2.1 체크리스트

체크리스트는 서비스 중 프로그램을 분석하거나 프로그래밍하기 이전에 매우 유용하다.

이곳에서 수집된 정보는 추후에 필요한 실제 데이터를 제공함으로써 빠르게 문제를 해결 할 수 있다.

그림 3: 체크리스트

프로그램을 분석하기 위한 수많은 방법들이 있다. 서로 다른 위치탐색방법과 전략 분석을 결합하면 에러의 위치를 검출할 때 상당한 효율을 발휘할 수 있다.

에러 위치 탐색 방법
다양한 툴 중에서 적합한 툴을 선택해야하므로, 에러를 검색할때 적절한 탐색 방법 선정은 매우 중요하다. 기계부터 시작해서 컨트롤러까지 아우르는 실제 환경에 적합한 연속적인 질문이 요구된다.

- 문제 분석
- 처리할 수 있는 에러 제거
- 신호 측정

개요에 따라, 명확하게 구분되고 섬세하게 분석될 수 있다.

환경과 일반적인 상태
그 문제가 기계와는 무관하지만 환경적인 원인으로 발생할 수 있기 때문에, 즉시 다양한 분석 전략을 적용하는것은 좋은 생각은 아니다.

런타임(쉬프트 또는 product/batch 변경, 클락 변경(예: 햇빛 저장시간), 방 온도, 센서 교체, 사용자 활동, 등등.) 동안 일반적인 상태를 살피면 에러 검색 범위를 좁힐 수 있다.

기계 환경에 잠재적인 에러를 제외하고, Automation Studio 프로젝트 스스로 분석을 시작한다.

한 번 발생한 에러 또는 반복적으로 발생하는 에러?
에러가 특정 동작에서 반복적으로 발생하다면, 그 에러는 디버거를 사용하여 프로그램 코드를 분석할 수 있다.

특정 상황에서 발생되는 프로그램 에러나 어떤 규칙에서 발생되는 에러가 아닐경우에는 재발생하기 어렵고 심지어 재생산되더라도 항상 신뢰하기 어렵다.

반복적으로 발생하지 않는 에러는 어플리케이션(예: Profiler 자동 활성화)에서 충분히 준비하면 쉽게 분석할 수 있다.

프로그램 또는 프로그램 시퀀스에서 발생한 에러?
런타임 에러는 프로세스를 실행 중에 사소한 것을 고려하지 않아 발생한다.

프로그램을 실행시 에러의 예:

- 0 으로 나눠짐
- 함수의 리턴 값을 구하지 못했을 때
- 배열이 오버플로(overflow)되었을 때(예: loop counters)
- 초기화되지 않은 포인터를 접근할 때

문제를 전달할 때 필요한 정보는 무엇인가?

문제를 해결하기 위해 추가적인 인력이 필요하다면, 그것에 대한 상세한 설명이 필요하다:

- 체크 리스트(다음페이지에 있는)와 관련된 상세한 설명
- 어떤 행동을 취했는가?
- 제외시킬 수 있는 환경이 무엇이 있는가?
- 직무 환경에서 문제를 재생산할 수 있는가?

상세한 행동에 대한 정보와 문서를 얻으면 문제를 발견할 가능성이 더 높아진다.(교육자료 “TM920-Diagnostics and service for end users”를 참고하라)

(업그레이드 버전을 포함한) 소프트웨어 버전

소프트웨어	버전	설명, 비고
●	●	●
●	●	●
사용하는 하드웨어(설치된 운영시스템을 포함한)		
모델 번호(Model number)	리비전(Revision) 시리얼 번호(Serial number)	설명, 비고
●	●	●
●	●	●
문제가 재발하는가, 또는 한번만 발생했는가?		
●		
어떤 동작에서 문제가 재발하는가?		
●		
문제가 언제 시작했는가? 소프트웨어나 하드웨어 구성 또는 기계 환경 변화가 있는 이후인가?		
●		
CPU 의 상태는 어떤가? 그리고 동반된 컴포넌트의 LED 상태는 어떤가?		
●		
문제를 분석하기 위해 CPU 에서 불러온 정보(스크린샷이 아닌)는 무엇인가? 예: logger, Profiler 데이터 등등		
●		

표 1: 정보 분석을 위한 체크리스트

2.2 진단 툴의 개요

Automation Studio 는 프로그래밍, 시운전 그리고 서비스 동안 진단을 위한 적절한 툴을 제공한다.

올바른 진단 툴을 선택해야만 정확하고 빠르게 필요한 정보에 접촉할 수 있다.

Automation Studio 도움말(Automation Studio help)은 개발, 시운전 그리고 서비스를 하는 동안 도움을 주고 다양한 진단 툴에 대한 상세한 정보를 제공한다.

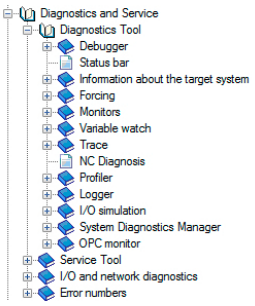

그림 4: 진단에 대한도움말

예제: 진단 툴에 대한 도움말 열기
진단 툴은 “diagnostics and servise” 섹션에 있다. 여기에서 전반적인 도움말 구성 개요를 알 수 있다.

Diagnositcs and service

이 교육자료는 서로다른 테스크 (task)를 사용하여 진단 툴을 실행한다.

3 시스템 정보 읽어오기

Automation Studio 에서 타켓 시스템의 시스템 정보를 읽어 올 수도 있고 인터넷 브라우저를 통해서도 시스템 정보를 확인 할 수 있다.

시스템 정보 읽어오기

3.1.1 “상태 바(Status bar)”	연결상태, Automation Runtime 버전, 컨트롤러의 동작 상태 등에 대한 정보
3.1.2 “타겟 시스템의 정보 (Information about the target system)”	메모리 정보, 배터리 상태, 컨트롤러의 날짜/시간 설정 옵션 등의 디스플레이 정보
3.2 “온라인 비교 (online comparison)”	프로젝트와 컨트롤러에 탑재된 소프트웨어 버전을 비교. 이 기능은 하드웨어에 이용할 수 있다. 프로젝트 안에 설정되고 컨트롤러에 실제로 존재하는 모듈을 비교할 수 있다.
3.3 “Logger 에서 에러 분석 (Error analysis in Logger)”	런타임 중에 타겟 시스템에서 발생하는 디스플레이 이벤트
6.1 “System Diagnostics Manager (SDM)”	System Diagnostic Manager (SDM)은 Automation Runtime 에 직접적으로 통합되어 있는 웹 기반 인터페이스이다. 표준 인터넷 브라우저는 중요한 타겟시스템 정보를 분석하는데 사용될 수 있다.

표 2: 시스템 정보 읽어오기

이번 단원에서 예제 관련 필요사항

필요한 하드웨어와 Automation Studio 프로젝트를 사용하여 아래에 예제를 진행한다.

이 장에서 사용되는 X20 CPU 기반의 프로젝트에 대한 설명과 이미지는 TM210(Working with Automation Studio)와 TM213(Automation Runtime)에서 설계된 X20 CPU 프로젝트를 참조한다.

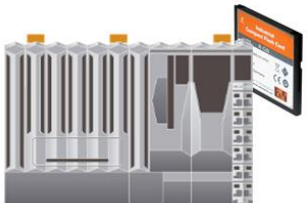

그림 5: X20 CPU

전제조건과 요구사항

- 컨트롤러에서 실행 가능한 프로젝트
- Automation Studio 와 컨트롤러의 온라인 연결

3.1 컨트롤러 동작 상태

Automation Studio 에서 컨트롤러의 동작상태를 판단하기 위해 아래의 옵션을 사용할 수 있다.

- 3.1.1 “상태 바(Status bar)”
- 3.1.2 “타겟 시스템의 정보(Information about the target system)”
- 3.2 “온라인 비교(online comparison)”

이 정보들은 System Diagnostics Manager (6.1 “System Diagnostics Manager (SDM)”)을 사용하여 볼 수 있다.

3.1.1 상태 바(Status bar)

상태 바는 Automation Studio 창 하단에 위치한다.

그림 6: 상태 바

상태 바는 아래의 정보들을 포함하고 있다:

1. 연결 설정
2. CPU 타입과 Automation Runtime 버전
3. 컨트롤러의 동작 상태("TM213 – Automation Runtime" 참조)

Project management ₩ The workspace ₩ Status bar
Real-time operating system ₩ Method of operation ₩ Operating status

3.1.2 타겟 시스템의 정보

온라인연결이 되면 메인메뉴의 <Online> / <Info> 또는 Pysical View 단축메뉴(마우스 오른쪽 버튼 클릭)의 <Online Information…>을 사용해서 타겟 시스템에 대한 정보를 확인 할 수 있다.

타겟 시스템의 시간은 직접 설정하거나 다이얼 로그 박스에 있는 PC 의 시간과 동기화시킬 수 있다.

"Info" 다이얼 로그 박스는 아래의 정보를 포함한다:

- 내부 백업 배터리의 상태 테스트
- 컨트롤러 종류와 Automation Runtime 버전
- 하드웨어 노드 넘버
- 타겟 시스템에서 사용가능한 메모리
- 타겟 시스템의 날짜와 시간

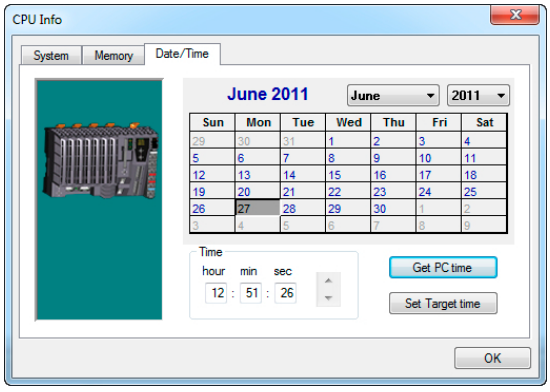

그림 7: 타겟 시스템의 날짜와 시간 설정

Diagnostics and service ₩ Diagnostic tools ₩ Information about the target system

3.2 온라인 비교(Online comparison)

개요 전에, 프로젝트의 하드웨어와 소프트웨어 설정이 타겟 시스템의 설정과 일치하는지 확인해야한다. 이 과정은 Automation Studio 의 비교 기능을 이용해서 할 수 있다.

3.2.1 온라인 소프트웨어 비교

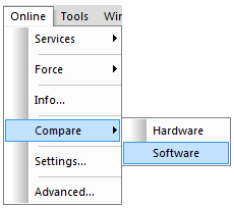

그림 8: 온라인 소프트웨어 비교 활성화

온라인 소프트웨어 비교는 타겟 시스템에 테스크(task)의 상태와 버전을 비교하고 프로젝트에서 소프트웨어 구성(software configuration)을 비교하는데 사용된다.

온라인 소프트웨어 비교는 메인메뉴의 <Online> / <Compare> / <Software>를 선택하면 열 수 있다.

아래의 정보가 분석된다:

- 프로젝트에 포함된 테스크(task)와 타겟 시스템의 테스크(task) 비교
- 테스크(task)에서 사용하는 타겟 메모리
- 개별적인 테스크(task)의 동작 상태
- 마지막 빌드 버전과 타임 스탬프

한 창에 두개의 세로줄(열)이 열린다. 왼쪽은 소프트웨어 구성(software configuration)에 설정된 요소들을 보여주고 오른쪽은 타겟 시스템에 활성화된 구성을 나타낸다.

이번 예제에서, 타겟 시스템에 있는 **"LampTest"** 테스크(task)는 중지되었지만, **"Loop1"** 테스크(task)는 타겟 시스템에 존재하지 않는다.

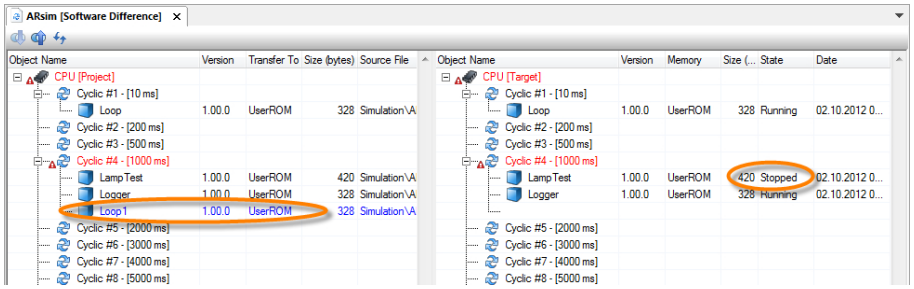

그림 9: 온라인 소프트웨어 비교

Diagnostics and service ₩ Diagnostic tools ₩ Monitors ₩ Online software comparison

3.2.2 온라인 하드에웨어 비교

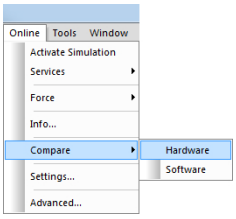

그림 10: 온라인 하드웨어 비교 창 열기

온라인 하드웨어 비교는 프로젝트에 있는 하드웨어 설정과 런타임때 실제 사용되는 하드웨어 설정을 비교한다. 온라인 하드웨어 비교는 메인메뉴의 <Open> / <Compare> / <Hardware> 를 선택하면 활성화 시킬 수 있다.

창이 두개로 나뉘어 진다. 왼쪽은 프로젝트에 있는 하드웨어 설정이다. 오른쪽은 런타임에서 사용되는 하드웨어 설정이다. 차이점이 빨간색 세모로 경고 표시 되어있다.

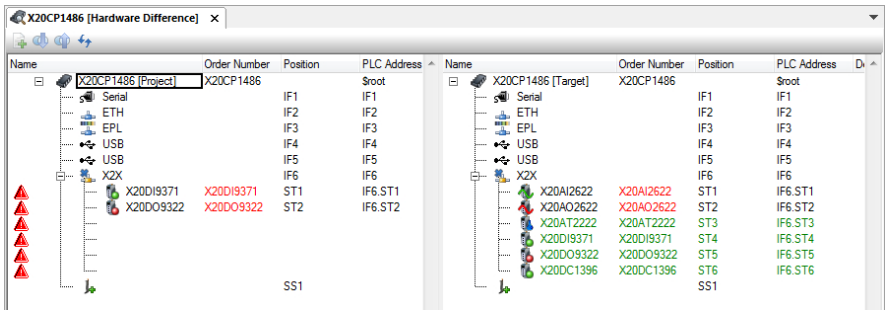

그림 11: 온라인 하드웨어 비교

이 설정에서, X2X 링크 버스(X2X Link Bus)에 두 디지털 모듈이 설정되었지만, 두 아날로그 모듈은 타겟 시스템에서 사용되고 있다. 식별된 다른 모듈들은 프로젝트에서 설정되지 않았다.

Diagnostics and service ₩ Diagnostic tools ₩ Monitors ₩ Online hardware comparison

3.3 Logger 에서 에러 분석

Automation Runtime 은 어플리케이션이 동작할 때 발생하는 모든 치명적인 에러(예: cycle time violation), 경고, 정보 메시지(e.g. 재시작) 등을 기록한다.

기록은 컨트롤러 메모리에 저장되고 재시작 후에 사용할 수 있다.

Diagnostics and service ₩ Diagnostic tools ₩ Logger window

- Opening the Logger window
- Operating the logger ₩ Storing ₩ Loading logger data
- User interface description ₩ Bactrace

3.3.1 온라인 연결 상태의 Logger

Logger 는 Physical View 에서 <Open> / <Logger>를 누르거나, 컨트롤러의 단축 메뉴에서 <Open Logger>를 누르거나, 키보드 단축키 <CTRL> + <L>을 사용해서 열 수 있다.

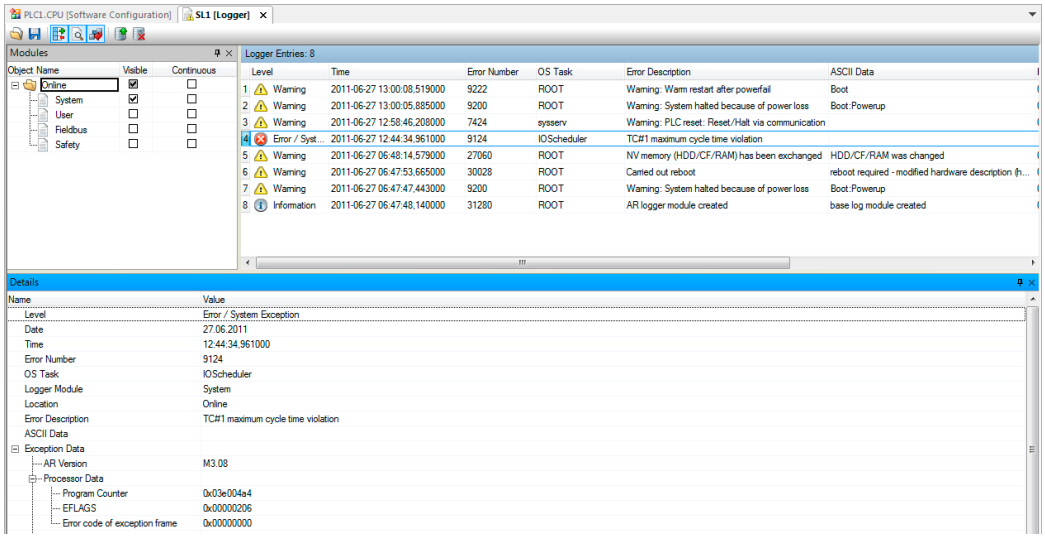

그림 12: Log window

위에 보이는 그림은 CompactFlash data 생성과 컨트롤러에 이 데이터를 로딩한 후, Automation Runtime 에 의해 기록된 이벤트을 보여준다.

예제: cycle time violation 발생과 Logger 목록 체크

"Loop" 프로그램(8.1 ""Loop" 샘플 프로그램")에 기초하여 cycle time violation 은 변수 모니터에서 "udiEndValue" 변수를 증가시키면 발생한다.

재시작 후 Automation Studio 와 타겟 시스템 사이에 온라인 연결이 완료 되면, Logger 창을 열고 서비스 모드에서 부팅된 원인을 찾는다.

1) "Loop" 테스크(task)의 변수 모니터(variable monitor)를 연다.
   4.1 "변수 모니터링과 수정"참조

2) "udEndValue" 변수 값을 cycle time violation 이 발생(연결 누락과 서비스 모드에서 재시작)할 때까지 증가시켜라.

3) Physical View 에서 Logger 연다.

4) 서비스 모드에서 부팅된 원인을 찾는다.

5) 목록을 선택하고 F1 을 누른다.

열리면, 서비스모드에서 부팅된 원인을 Logger 에서 확인할 수 있다.

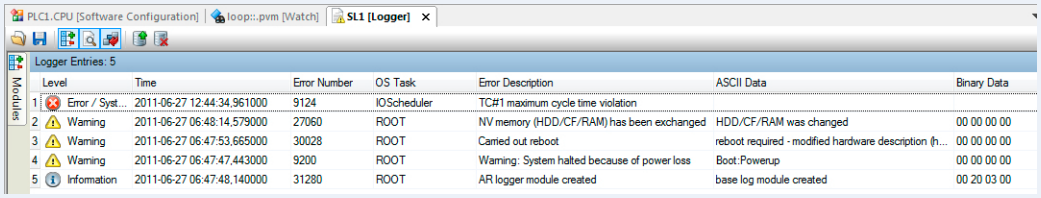

그림 13: Logger 에서 cycle time violation

Logger 에서 항목을 선택하고, <F1>을 누르면 Automation Studio 도움말에서 에러에 대한 상세한 설명을 볼 수 있다. 에러 항목에 대한 추가정보는 백트레이스를 보면 알 수 있다.

아래의 그림은 Automation Studio 도움말에서 선택된 에러 번호(Error number)에 대한 설명이다.

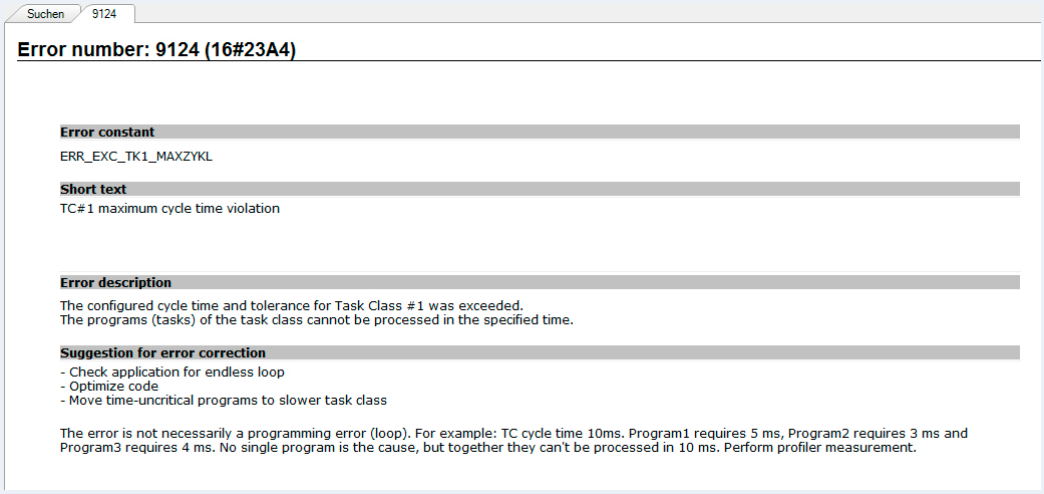

그림 14: Automation Runtime 에러에 대해서 세심하게 전후 맥락이 적혀있는 도움말

3.3.2 Logger 데이터의 오프라인 평가

Logger 기록은 컨트롤러와 연결되지 않아도 평가할 수 있다.

그 데이터는 Automation Studio 에 있는 온라인 연결이나 System Diagnostics Manager 에 의해 항상 업로드 된다.

어플리케이션 케이스(Application case)

Logger 데이터는 System Diagnostics Manager 를 사용해서 서비스 엔지니어에 의해 저장된다. 옮겨진 데이터는 Automation Studio 에서 열리며 분석된다.

Automation Studio 에서 Logger 항목이 저장되고 Logger 툴바를 사용해서 다시 열수 있다.

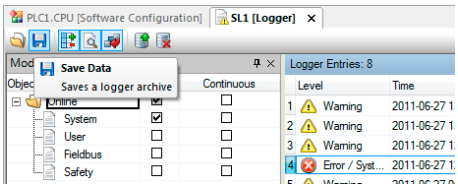

그림 15: Logger 항목들의 저장

3.3.3 사용자 log 데이터 생성하기

Logger 기능은 어플리케이션에서 발생한 이벤트를 기록하기 위해 어플리케이션 프로그램에서도 사용할 수 있다.

ArEvenLog 라이브러리의 기능을 이용해서 할 수 있다. 이 기능에는 32-bit 이벤트 ID 가 들어간다. 이것은 시스템 내에서 뚜렷이 구분된다. 에러, 경고, 정보, 성공으로 구분된다. 사용자 이벤트 ID(Event ID)와 시스템 이벤트 ID 는 쉽게 구분된다.

사용자는 시스템에서 사용되는 어떤 이벤트 ID 도 입력할 수 없다. 이는 라이브러리에 의해 보호된다.

32-bit 이벤트 ID(Event ID)는 아래와 같이 입력한다.:

Bit 31~30	Bit 29	Bit 28	Bits 27-16	Bits 15-0
Severity	1 .. Customer	Reserved	Facility	Code

사용자 이벤트 ID 는 해당되는 회로 다이어그램에 따라 생성되거나 ArEventLogMakeEventID() 함수를 이용해서 생성한다.

이 기능은 12-bit 가 확실하게 장소를 찾을 수 있도록 해준다. 사용자 이벤트 ID 에서 이 기능은 다음과 같이 나눠진다:

- 0.. 15 : 고객 어플리케이션
- 16.. 4096 : 장비 생산과 특별한 경우를 위한 장소

Programming ₩ Libraries ₩ Configuration , system ₩ ArEventlog

적용:

- 서비스 활동 저장하기 (예: 배터리 교체)
- 사용자 활동 저장하기 (예: 금지된 항목)
- 예외 테스크(task) 이벤트 검색하기 그리고 Logger 에 기록하기
- 비활성화된 모듈 모니터링(e.g. modulOK = FALSE) 이벤트 저장하기

예제: 사용자 Log 데이터 생성

"userlogger"라는 이름의 사용자 logbook 을 Automation Studio 프로젝트에 생성하라. **"This is a user warning"**이라는 메시지를 Logger 에 입력라라. 이는 warning 을 수반한다 (severity = warning). ArEvent-Log 라이브러리를 사용라라.

1) ArEventLogCreate()를 사용해서 사용자 logbook "userlogger"를 생성하라.

2) ArEventLogGetIdent()를 사용해서 사용자 logbook 의 ident 를 읽어라.

3) 함수 ArEventLogMakeEventID()를 사용해서 32-bit 이벤트 ID 를 생성하라.

4) ArEventLogWrite()를 사용해서 사용자 logbook 에 Logger 항목을 입력하라.

5) 사용자 logbook 에 생성된 Logger 항목과 이벤트 ID 를 확인하라.

4 프로세스 변수 모니터링과 분석

프로세스 변수(Process variables)는 Automation Studio 에서 다양한 방법으로 모니터링되고, 분석하고, 수정된다.

프로세스 변수 모니터링과 분석

4.1 “변수 모니터링과 수정”	Watch 창은 타겟 시스템의 변수 값을 보고, 모니터링과 수정을 지원하며 한 축 (NC Watch)의 현재 상태를 볼 수 있다.
4.2 “실시간 변수 기록하기”	추적(trace) 기능은 정해진 주기를 넘어서 실시간으로 몇개의 변수 값을 기록할 수 있다. 이 데이터는 Automation Studio 를 사용해서 업로드 되고 커브 형태로 보인다. NC 추적 기능은 실시간 데이터가 드라이버에 직접 저장된다.
4.3 “I/O 모니터링과 force”	I/O 모니터는 I/O 값을 분석하고 사용하지 않는 I/O 채널과 네트워크 품질까지도 분석한다.
5.2 “소스 코드에서 에러 검출”	광범위한 진단 기능은 텍스트 기반과 비주얼 프로그래밍 언어 둘 모두 사용 가능하다.

표 3: 프로세스 값의 모니터링과 분석

이번 단원에서 예제 관련 필요사항

이번 단원의 설명과 그림들은 Automation Runtime simulation (ARsim)을 사용하는 Automatio Studio 프로젝트 “CoffeMachine”을 참조한다.

- “CoffeMachine” 프로젝트를 Automation Runtime simulation (ARsim)에 전송하라
- Automation Studio 와 Automation Runtime simulation(ArSim) 사이의 온라인 연결

그림 16: ArSim

NC 데이터 분석에 관련된 자료는 모션 교육 자료 (TM4xx)에서 확인 할 수 있다.

4.1 변수 모니터링과 수정

Watch 창에서 타겟 시스템의 변수 값을 모니터링하고 수정할 수 있다.
변수 리스트는 추후에 진단, 함수 테스트 등에 재사용될 수 있도록 변수 모니터에 저장된다.

예제: “CoffeMachine” 어플리케이션 작동시키고 진단하기

Automation Studio 에서 변수 모니터를 사용해서 **“CoffeMachine”** 어플리케이션을 작동시켜라.

아래의 표를 보고 변수 모니터로 변수을 입력해라. 변수 값을 조작하고 모니터링하며 **“CoffeMachine”** 어플리케이션의 프로세스 시퀀스를 테스트하라.

만약 테스크(task) 클래스의 설정된 런타임이 초과된다면, 테스크(task) 클래스의 다음 시작이 원래 설정된 사이클의 시작으로 쉬프트된다. 이는 어플리케이션의 타이밍 문제를 발생시킬 수 있다.

이 테스크에서 아래의 프로세스 변수가 요구된다:

변수 활동 범위	프로세스 변수	설명
커피 타입 0-2	gMainLogic.par.coffeeType	선택된 레시피 변경
레시피 0-100	gMainLogic.par.receipe.coffee gMainLogic.par.receipe.milk gMainLogic.par.receipe.sugar gMainLogic.par.receipe.water	이 매개변수들은 선택된 커피의 타입(0,1,2)에 대한 레시피를 변경하는데 사용된다.
커피 가격 0.0-10.0	gMainLogic.par.receipe.price	선택된 레시피의 커피 가격
지불 0-10	gMainLogic.par.givenMoney	커피의 가격과 비교되는 주입된 돈
스위치 on/off 0-1	gMainLogic.cmd.switchOnOff	스위치 on 후에, 준비과정이 반드시 필요하다. 물 온도를 체크하라.
절차 1	diStartCoffee	선택된 음료 준비를 시작하기 위한 명령어
물 온도	Gheating.status.actTemp	물 온도는 선택된 커피 타입에 따라 조절된다.
메시지	gMainLogic.cmd.vis.messageIndex	준비 과정과 지정된 온도에 도달하는 동안 이 곳에 메시지 인덱스가 보여진다.
프로세스 시퀀스	gMainLogic.status.progressStep	커피가 준비되는 동안, 과정이 보인다. Value = 1 이면 꽉참. Value = 2 이면 컵을 내놓는다.

1. Watch 창에 프로세스 변수 추가하기

- “CoffeeMachine” 프로젝트를 Automation Runtime simulation (ARsim)으로 전송하라.
- 소프트웨어 구성에 있는 “mainlogic”에서 변수 모니터 (Watch)를 열어라.

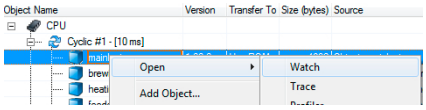

그림 17: 변수 모니터 (Watch) 열기

- 툴 바를 사용하거나 <Ins> 키를 눌러서 표에 있는 변수을 입력하라.

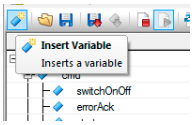

그림 18: 변수 입력

변수 모니터(Watch)에 변수가 입력되면, 어플리케이션 프로세스 시퀀스가 시뮬레이션 된다.

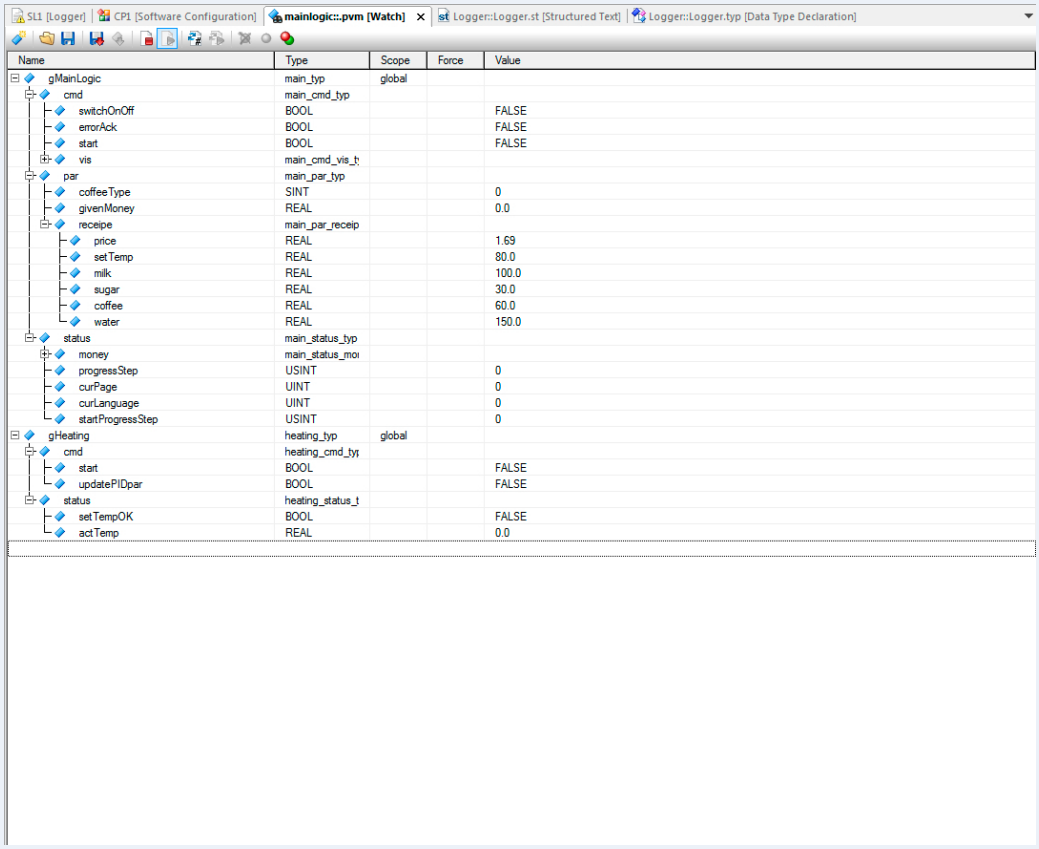

그림 19: 변수 모니터(Watch)에서 변수 보기

2. 커피 머신 구동시키고 Watch 창 에서 동작시키기

1) 스위치 온
   "gMainLogic.cmd.swtchOnOff=1"
2) 프로세스 상태 체크
   온도가 설정한 값에 이르면 **"gMainLogic.cmd.vis.cessageIndex = 2"**가 된다.
3) 입금
   입금된 금액("gMainLogic.par.givenMoney")이 커피 값("gMainlogic.par.receipe.price")과 같거나 커야한다.

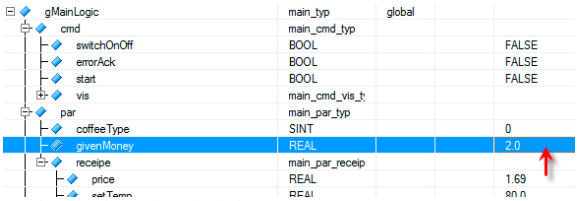

그림 20: 입금 시뮬레이션

4) 준비 단계 시작
   "diStartCoffee = 1"
5) 진행 과정 모니터링
   "gMainLogic.status.progressStep" 변수를 모니터링해라.
   컵에 커피가 꽉 차면 Value = 1 , 컵을 내놓으면서 Value = 2 가 된다.

변수 모니터에 있는 변수 목록은 추후 사용을 위해 저장된다. 따라서 언제든지 프로세스 시퀀스가 재실행 될 수 있다.

컨트롤러에 있는 변수들은 Watch 창에서 모니터링되고 수정된다.
이 변수들에 대한 관찰, 데이터 타입과 I/O 타입 같은 추가적인 정보도 볼 수 있다.
다양한 테스크를 관리하기 위하여 별도의 리스트로 변수를 관리할 수 있다.

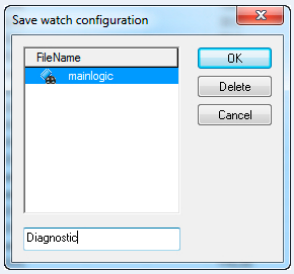

그림 21: 변수 리스트 저장하기

Diagnostics and service ₩ Diagnostics tool ₩ variable watch

4.1.1 동시에 변수 쓰기

Watch 창에서 변수를 변경하면, <Enter>를 누르는 즉시 컨트롤러에 전달된다.

컨트롤러는 다음 사이클에 새로운 값을 적용한다.

즉시 적용되지 않고 변수 모니터에 값을 입력하기 위해서는, archive 모드를 켜야한다.

Archive 모드는 툴바에서 “Archive mode” 아이콘을 사용해서 시작하고 종료할 수 있다.

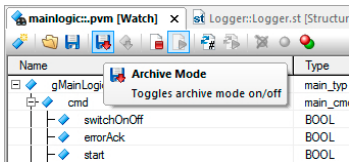

그림 22: archive 모드 시작하기

수정이 필요한 변수 값을 변수 모니터에 입력한 뒤, 툴바에 있는 “Write values” 버튼을 클릭하면 컨트롤러에 변경된 값 전부를 보낼 수 있다.

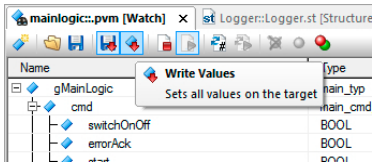

그림 23: archive 모드에서 모든 변수 변경하기

동시에 입력할 때 변수 모니터에 입력되는 변수에 대해 신중하게 생각해야 한다. archive 모드를 정확하게 사용하지 않는다면 프로세스 시퀀스에 변경사항이 있을 때 잘못된 실행결과가 나타날 수 있다.

Diagnostics and service ₩ Diagnostics tool ₩ Watch window ₩ Archive mode

4.2 실시간 변수 기록하기

Automation studio 변수 모니터에서 컨트롤러 변수(controller variables)는 비동기[1]로 읽혀진다.

Automation Runtime 테스크(task) 클래스 시스템에서 실제 값의 비동기 접근 타입은 아래의 제한조건에서 변한다.:

- 테스크 클래스 에 비동기로 값을 보여줌.
- 일련의 값 변경과 그 변수에 속해있는 값들을 결정할 수 없음.

"Trace" 기능은 테스크(task) 클래스의 맥락 안에서 타겟 시스템에 변수 변화를 기록할 수 있다.

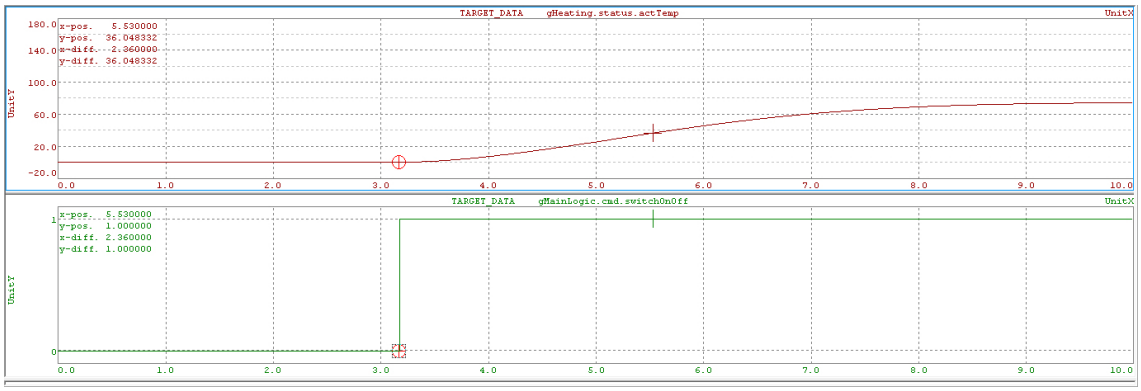

그림 24: Track 기록의 예

이 예시는 변수가 특정 상태에 도달되었을 때 다른 과정이 어떻게 시작되는지를 보여준다. 측정 커서는 두 커브의 시간에 따른 값 변화를 확인하는데 사용한다.

기록과 어플리케이션 과정을 분석함으로써 최적화 시키고 에러를 검출 할 수 있다.

[1] 여기서 비동기화는 정해진 스케줄에 따라 네트워크를 통하여 변수 값이 변수 모니터에 전달되지 않았을 때를 의미한다.

Trace 다이얼로그 박스는 소프트웨어 구성에서 상응하는 테스크(task)를 선택하고 마우스 우측을 클릭, <Open> / <Trace> 를 사용해서 열 수 있다.

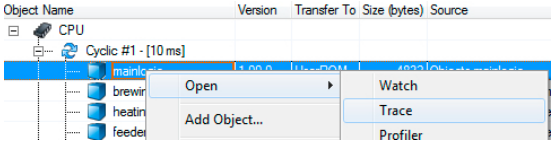

그림 25: 소프트웨어 구성에서 Trace 창 열기

새로운 Trace 구성은 "Insert Trace Configuration" 버튼을 눌러서 Trace 다이얼로그 박스에 입력한다. 기록되야 하는 변수들은 "Insert New Variable"을 클릭하면 Trace 구성에 추가된다.

예제: 변수에 따라 변화는 커브 기록하기

"CoffeMachine" 프로세스 시퀀스에서, 과정이 시작되면 물 온도는 따뜻해지는 과정을 거친다. 커피 타입에 따라 물 온도 목표치가 변한다; 물 온도는 목표치에 도달할 때까지 계속해서 확인한다.

테스크는 온도 정보를 실시간 기록함으로써 손쉽게 분석할 수 있는 물 온도 규제 방법을 보여준다(이 경우에는 뚜렷한 오버슈트가 있다).

아래의 프로세스 변수들이 필요하다:

Action	변수 범위	프로세스 변수
커피 타입 선택	0-2	gMainLogic.par.coffeeType
스위치 on/off	0-1	gMainLogic.cmd.switchOnOff
물의 온도	-	gHeatine.status.actTemp

1. Trace 창 열고 변수 추가하기

- "mainlogic" 테스크에 대한 Trace 창을 열기
- 새로운 Trace 구성 입력
- 기록을 위해 필요한 프로세스 변수 입력
  Trace configuration 은 아래와 같다:

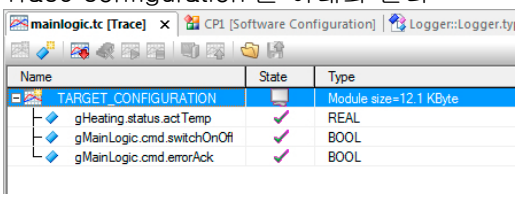

그림 26: Trace configuration

값 들은 테스크 클래스에 따라 반복해서 기록된다. 기록 기간과 시작 상태는 Trace 구성의 속성(properties)에서 설정한다.

이번 에제에서, coffee machine 스위치가 on (**gMainLogic.cmd.switchOnOff = 1**)이 되면 기록을 시작한다.

2. trace 구성: recording buffer 와 trigger condition 을 설정해라

- Trace 구성의 속성(properties) 다이얼로그 박스를 열기

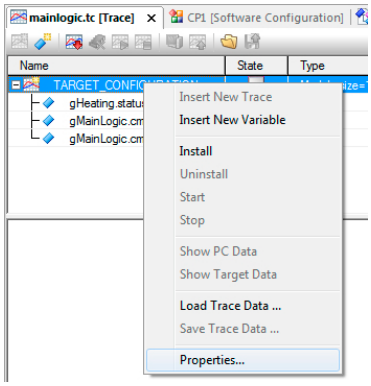

그림 27: Trace 특성

- Recording buffer 설정
  Recording buffer 크기는 "General" property page 에서 30000 로 설정
- Trace 모드 변경
  기록을 시작하기 위한 Trigger 상태는 "Mode" property page 에서 설정 (gMianLogic.cmd.switchOnOff = 1).

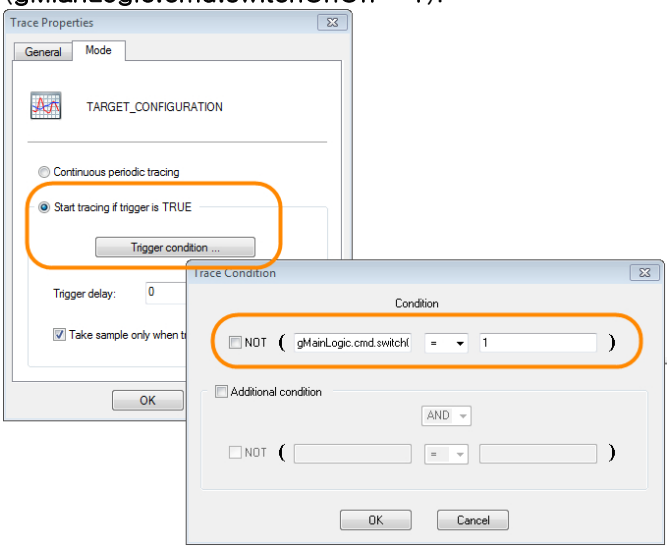

그림 28: trigger 상태 설정

trigger 상태 변수를 선택하기 위한 다이얼로그 박스는 <Spacebar>를 누르면 열 수 있다.

기록(recording)이 설정되면, "Install"을 클릭하면 타겟 시스템에 전달된다.

이 경우에, 툴바에 있는 "Start" 아이콘을 직접 눌러도 기록이 실행되지 않지만 시작 조건(start condition)이 충족되면 기록이 실행된다.

3. 프로세스 시작과 Trace 데이터 분석

- Watch 창 열기
  변수 모니터를 열고 위의 표에서 필요한 변수 입력
- Coffee machine 을 작동 시키기
  “gMainLogic.cmd.switchOnOff” 변수 설정
- 커피 타입 변경
  “gMainLogic.par.coffeeType” 을 0, 1, 2 로 변경하라. 커피 타입 변경으로 인해 물의 설정 온도 값도 변경된다. 실제 물의 온도 변화는 Trace 창에 기록된다.

테스크에서 Watch 창 설정이 저장되면, 자동으로 다시 열린다.

툴바에서 “Stop”아이콘을 클릭하면 언제든지 기록을 멈출 수 있다. 업로드 후에 “Show target data” 아이콘을 클릭하면 그 결과를 볼 수 있다.

시작 조건(start condition)이 충족됐을 때 데이터가 기록된다. 변수 모니터에서 필요에 따라 값을 수정할 수 있다. 타겟 시스템에서 데이터가 업로드된 후에, 아래와 같이 기록이 보인다(변수 값이 어떻게 변경되는지).

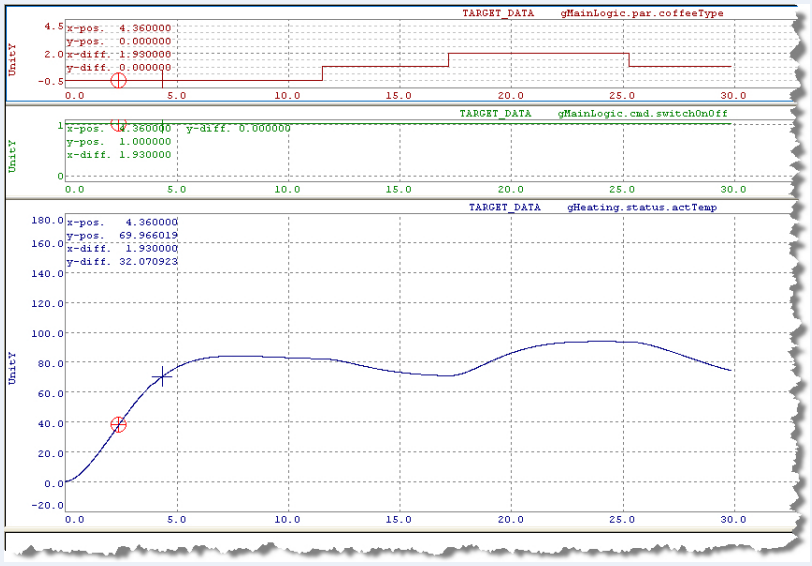

그림 29: Tarce 데이터

오버 타임에서 값의 변화는 측정 커서를 사용해서 분석할 수 있다. 이는 시간축의 모든 변수에 대해서 똑같이 분석 할 수 있다.

Diagnostics and service ₩ Diagnostics tool ₩ Trace window

4.3 I/O 모니터링과 force

그림 30: 모니터 모드 ON

Physical View 에서 모듈을 더블 클릭하면 I/O 맵핑 창이 열린다. 온라인 연결이 활성화 되어 있고 적절한 모니터 모드를 선택했다면 I/O 채널의 Physical 상태가 보인다.

"Force" 옵션은 그 채널에 대해 (실제 physical 값에 상관없이) I/O 데이터 포인터를 할당 할 수 있도록 도와준다. 예를 들어 프로그램 시퀀스를 테스트를 위해서 사용된다. "Force" 기능은 I/O 데이터 포인트에 연결되는 변수 모니터와 I/O 할당에서 사용할 수 있다.

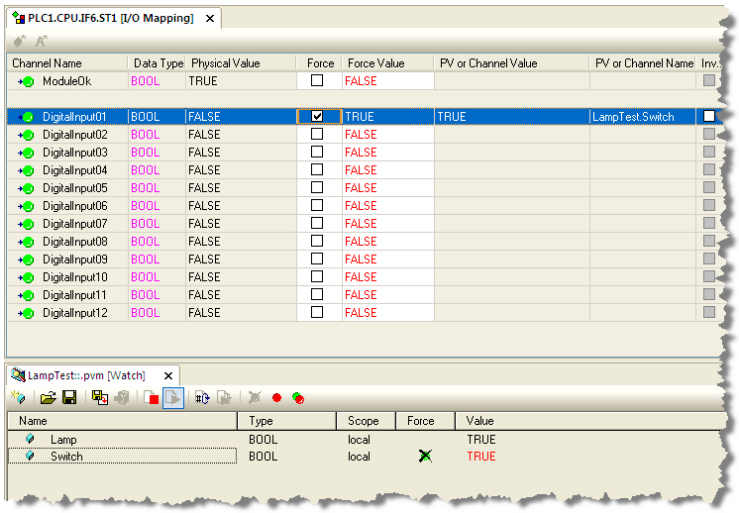

그림 31: 모니터 모드에서 I/O configuration; I/O configuration 과 변수 모니터에서 I/O 채널 forcing

입력 모듈에 I/O 데이터 포인트 강제할당(forcing)하기

입력 카드(예: X20DI9371) 채널에 "Force" 값은 Automation Runtime 에서 "시뮬레이션"된 값이다. 어플리케이션 프로그램 프로세스 시퀀스는 "force" 값을 받아서 작동하며 실제 인풋 상태 값을 사용하는 것은 아니다.

출력 모듈에 I/O 데이터 포인트 강제할당(forcing)하기

출력 카드 (예: X20DO9322) 채널에 "Force" 값은 어플리케이션 프로그램이 입력해왔던 값과 상관없이 해당하는 하드웨어 출력에 직접 입력된다.

시스템 시운전이 완료되면, force 옵션 효과는 필요가 없어진다. 시스템 **재시작(restarting)** 또는 메인메뉴에서 **<Online> / <Force> / <Global force off>**를 사용하면 자동으로 해제된다.

Diagnostics and service ₩ Diagnostics tools ₩ Monitors ₩ Mapping I/O channels in monitor mode
Diagnostics and service ₩ Diagnostics tools ₩ Force
Diagnostics and service ₩ I/O and network diagnostics

5 프로그래밍 작업중 소프트웨어 분석하기

Automation Studio 에는 어플리케이션 소프트웨어 설계에 도움을 줄 수 있는 진단 툴들이 있다.

뿐만 아니라 Automation Runtime 과 실제 소스 코드에서 어플리케이션과 소프트웨어 에러를 찾을 수 있는 방법이 있다.

프로그래밍 작업중 소프트웨어 분석

5.1 Profiler 설정과 데이터 평가	Profiler 는 테스크 런타임, 시스템과 스텍 로드 등의 중요한 시스템 정보를 보여주고 측정할 수 있다.
5.2.3 “Line coverage”	Line coverage 는 현재 작동하는 소스코드 라인을 가리킨다.
5.2.5 “소스 코드 디버깅”	Debugger 는 프로그램이나 라이브러리의 소스코드 에러를 찾기 쉽게 해준다.
5.2.6 “상태 변수와 리턴 값 평가하기”	상태 변수(status variables)는 어플리케이션에서 함수 호출 에러 또는 함수 호출 상태를 평가하는데 사용된다.
5.3 “프로그램에서 변수 사용하기”	출력 창은 진행중인 과정에 대한 정보를 보여준다. 예를 들어 빌드, 다운로드, cross-reference 목록 생성, 검색 결과 표시 등

이번 단원에서 예제 관련 필요사항

예제에는 하드웨어와 Automation Studio 프로젝트가 필요하다.

이번 단원에 관련된 설명과 이미지는 TM210(Working with Automation Studio)와 TM213 (Automation Runtime)에서 사용한 X20 CPU 프로젝트를 참조한다.

- 컨트롤러에서 실행 가능한 프로젝트
- Automation Studio 와 컨트롤러의 온라인 연결

그림 5: X20 CPU

5.1 Profileer 설정과 데이터 평가

Automation Runtime 은 런타임 환경을 자동으로 기록하도록 설정할 수 있다.

Profiler 는 Phsical View 의 CPU <Configuration>을 선택하면 설정할 수 있다.

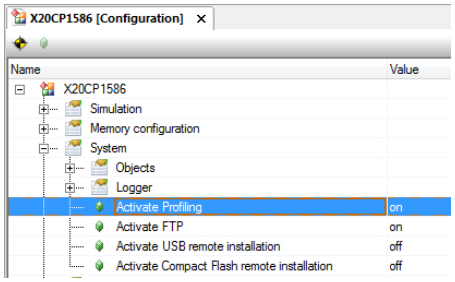

그림 33: Physicl View 에 보이는 CPU 특성에서 Profiler

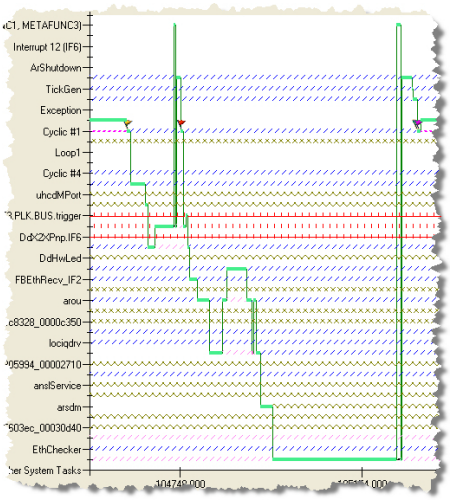

그림 34: profiling 예시

에러 (예: cycle time violation)가 발생한다면, 언제든지 타겟 시스템으로부터 기록을 받을 수 있고 Auto-mation Studio 에서 분석한다.

Diagnostics and service ₩ Diagnostics tools ₩ Profiler

5.1.1 Profiler 설정

Profiler 는 소프트웨어 configuration 의 <Open> / <Profiler>를 선택해서 열 수 있다.

Profiler 의 설정 다이얼로그 박스는 툴바의 "Configuration"아이콘을 클릭하면 열 수 있다.

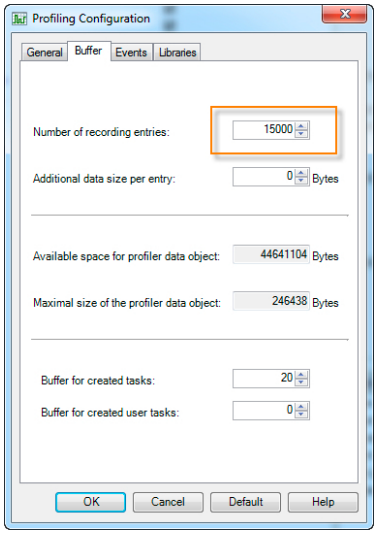

그림 35: Profiler 에서 사용될 number of recording entries 설정

기록할 때, "Events" 탭 아래에 있는 모든 이벤트을 기록하는 것을 권장한다. 이는 나중에 에러가 발생하거나 Profiler 데이터가 다른 누군가에게 제공할 때 필터링 할 수 있도록 해준다.

Profiler 설정 변경은 툴바의 "Install" 아이콘을 클릭하면 타겟 시스템에 전달할 수 있다.

Diagnostics and service ₩ Diagnostics tools ₩ Profiler ₩ Preparing the Profiler

5.1.2 Profiler 데이터 분석하기

Profiler 데이터는 순환 프로그램의 런타임 분석을 하기 위해 타겟시스템에서 Automation Studio Profiler 로 업로드된다.

예제: cycle time violation 을 발생시키고 Profiler 데이터 평가하기

"Loop" 테스크에서 "udEndValue" 변수 값을 증가시키면 Cycle time violation 이 발생한다.

타겟시스템을 서비스 모드에서 재시작한 후, Profiler 를 열고 타겟시스템으로부터 Profiler 데이터를 로드해라.

1)변수 모니터에서 "udEndValue" 변수 값을 500000 로 설정하면 cycle time violation 이 발생한다.

2)서비스 모드에서 재시작 시킨 후, 메뉴의 <Open> / <Profiler>를 선택해서 소프트웨어 configuration 의 Profiler 를 열어라.

설정된 cycle time + tolerance 가 런타임 동안 초과되었다면, Automation Runtime 은 exception 을 실행한다. 어플리케이션 프로그램이 exception 을 관리할 수 있도록 설정되어 있지 않다면, 타겟 시스템은 서비스모드에서 재시작된다.

Profiler 에서, 툴바의 "Upfoad data object"를 클릭하면 데이터가 업로드된다. 에러가 발생한다면, 재시작 될 때 새로운 Profiler 파일이 생성된다. 업로딩 과정 동안 리스트로부터 알맞는 파일을 선택할 수 있다.

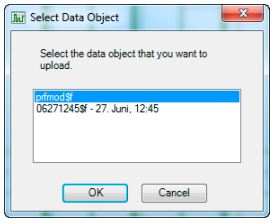

그림 36: Profiler 데이터 선택

툴바에서 "Zoom" 버튼을 누르면 Profiler 데이터가 보여지는 범위를 설정할 수 있다. 데이터 분석을 할 때, 100%에서 시작하는 것을 권장한다. <ESC>을 누르면 간단히 실행시킬 수 있다. 스크린에 가능한 많이 보여질 수 있도록 Project Explorer 는 숨겨놓을 수 있다.

Profiler 데이터는 보여지는 이벤트를 제한하기 위해 필터링할 수 있다.

어떤 이벤트들은 상황에 따라 보여야한다. Cycle time violation 의 원인을 찾을 때, 오른쪽에 보이는 그림과 같이 데이터는 필터링 될 수 있다.

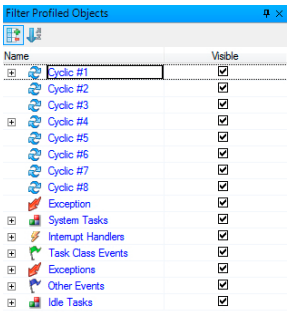

그림 37: Profiler 데이터 필터링

Diagnostics and service ₩ Diagnostics tools ₩ Profiler ₩ Recording Profiler data ₩ Analyzing Profiler data

어떤 시간에서 (예: 루프 사이클이 굉장히 많이 발생할 때), 테스크를 완료하는데 걸리는 시간이 사이클 타임과 허용 용량의 설정을 초과할 경우, 이 이벤트(exception)는 적당한 아이콘으로 나타난다.

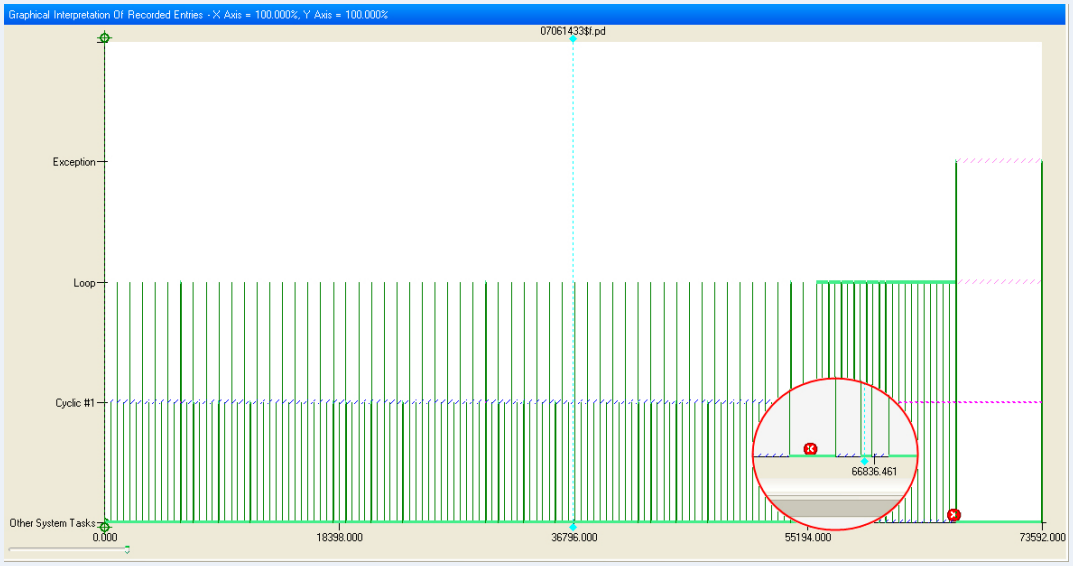

그림 38: Profiler 데이터에서 exception

원인 분석을 위해, 포인트 이전에 오는 데이터를 관찰해야 한다.

측정 커서를 사용하는 것과 Profiler 데이터에서 필요한 부분은 확대하는 것은 데이터를 분석하는 방법이다.

“Loop” 테스크 실행 시간

다음 그림에서 볼 수 있듯이, “Loop” 테스크는 보통 몇 microseconds 이내에 실행을 끝낸다(파란색 화살표); cycle time violation 은 사이클 타임(cycle time)과 허용 용량의 합이 초과(빨간색 화살표)되도록 실행된다면 발생한다.

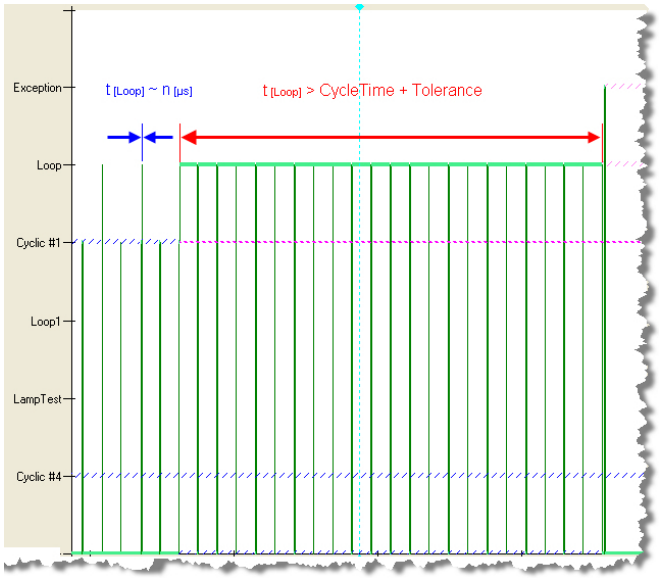

그림 39: cycle time violation 결정하기

아래 그림은 Profiler 에 간단한 어플리케이션이 저장되는 방법을 보여준다. 실제 어플리케이션에서는 몇 개의 테스크 / 테스크 클래스(Task Class)가 작동하고 있기 때문에 문제의 원인을 찾기가 힘들다.

예제:
두 테스크가 테스크 클래스 #1(TC #1)에서 동작중이며 일반적으로 구성된 테스크 클래스 사이클동안 수행되고 끝난다.

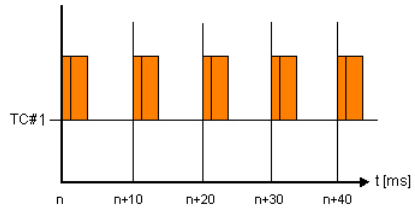

그림 40: 타이밍 다이어그램

첫 번째 테스크 완료 시간이 소요되고 (다이어그램에서 n+30 ms 이상) 두 테스크(task) 완료 시간이 사이클 타임과 허용 용량의 합을 초과한다면, 직접적인 cycle time violation 의 이유가 아니더라도 두 번째 테스크가 에러의 원인으로 지목될 것이다.

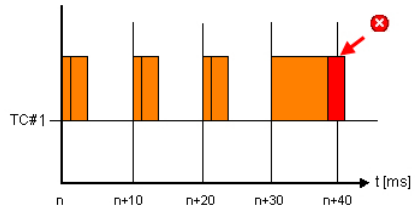

그림 41: 타이밍 다이어그램

이벤트 시퀀스는 raw data (툴바에 있는 “Output data” 아이콘) 평가에 따라 연대순으로 분석될 것이다.

Raw Data Of Recorded Entries

Nr.	Name	Event	Event Description
465	TickGen	0x000...	'TickGen' is now running - 'IOScheduler' was waiting
466	Cyclic #1	0x001...	'Cyclic #1' is now running - 'TickGen' was waiting
467	Cyclic #1	0x1e0...	Task class 'Cyclic #1' started
468	Cyclic #1	0x1e0...	Input scheduler of task class 'Cyclic #1' finished
469	Loop	0x020...	Cyclic task 'Loop' started
470	Loop	0x020...	Cyclic task 'Loop' finished
471	Cyclic #1	0x1e0...	End of cyclic programs in task class 'Cyclic #1'
472	Cyclic #1	0x1e0...	Output scheduler of task class 'Cyclic #1' started
473	DdX2XAcc.IF6	0x007f...	'DdX2XAcc.IF6' is now running - 'Cyclic #1' was waiting
474	tEpIV2If.IF3	0x007f...	'tEpIV2If.IF3' is now running - 'DdX2XAcc.IF6' was ready
475	DdX2XAcc.IF6	0x007f...	'DdX2XAcc.IF6' is now running - 'tEpIV2If.IF3' was delayed and waiting

그림 42: Profiler 기록에서 Raw data

이 목록은 “Loop” 테스크의 시작과 끝 항목을 보여준다. 연대순 시퀀스가 조금 더 추적한다면, 테스크가 스스로 시작하고 아직 끝나지 않았다는 것을 확인 할 수 있을 것이다.

5.1.3 어플리케이션 성능

Profiler 는 CPU 에서 테스크 성능 분석에 활용된다.

Profiler 데이터의 테이블 뷰는(적절히 필터링 된) 실행 시간과 각 테스크의 CPU 로드를 보여준다.

다음 그림은 툴바에서 **“Table”** 아이콘을 누르면 열 수 있다.

Name	CPU Usage [%]	Tolerance Count	Object Priority	Call Count	Minimal Net Time [µs]	Average Net Time [µs]	Maximal Net Time [µs]	Minimal Gross Time [µs]
Cyclic #1	2.145		230		217.356	218.539	219.832	965.138
loop	1.943		230	15	201.461	202.663	203.198	201.461
Cyclic #4	0.142		198		216.306	216.306	216.306	216.306
System Tasks	6.059							
Interrupt Ha...	1.061							
Idle Tasks	90.593							

그림 43: Profiler 을 사용한 CPU 로드 분석

Diagnostics and service ₩ Diagnostics tools ₩ Profiler ₩ Recording Profiler data ₩ Analyzing Profiler data ₩ Analyzing Profiler data in table form

5.2 소스 코드에서 에러 검출

통계적으로 소프트웨어 소스 코드 1000 줄에 두 개에서 세 개의 에러가 있다.

Automation Studio 는 프로그램 안에 소스 에러를 찾기 위해 광범위한 진단 툴을 제공한다.

5.2.1 프로그램 에디터에서 모니터 모드

모니터 모드는 프로그래밍 에디터 툴바에서 **“Monitor”** 아이콘을 선택하면 시작할 수 있다.

모니터 모드는 다양한 프로그래밍 언어에서 사용할 수 있고 몇 가지 방법으로 변수를 볼 수 있게 해준다:

그림 44: 모니터 모드 활성화

텍스트 기반 프로그래밍 언어 툴 팁
텍스트와 이미지 기반 언어에서 변수 값을 툴팁(tooltip)으로 볼 수 있다.

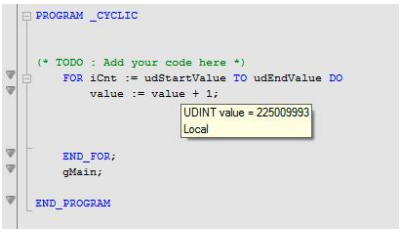

그림 45: 소스 코드안의 툴 팁

이미지 기반 프로그래밍 언어에서 변수 보기
이미지 기반 프로그래밍 언어에서 직접 변수 값을 볼 수 있다.

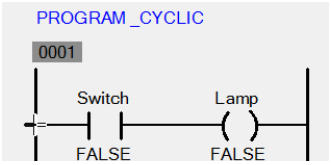

그림 46: 모니터 모드에서 이미지 기반 프로그래밍 언어

Watch 창
Watch 창은 모니터 모드가 활성화 됐을 때 소스코드 옆에 보인다. 소프트웨어 구성 또는 온라인 소프트웨어 비교에서 테스크를 선택하고 마우스 오른쪽을 누르면 Watch 창을 열 수 있다.

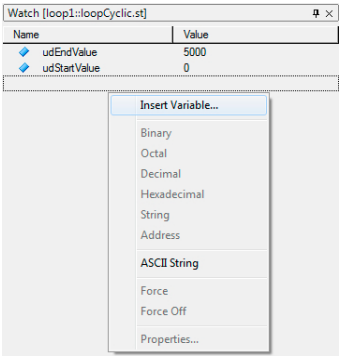

그림 47: 변수 모니터 창

5.2.2 Powerflow

이미지 기반 프로그래밍 언어, Ladder Diagram 으로 신호가 지나가는 길을 볼 수 있다. 신호가 지나가는 길을 툴바에서 **"Powerflow"** 아이콘을 선택하면 활성화 시킬 수 있다.

예제: Ladder Diagram 에서 Powerflow

"LampTest" 프로그램에서 Powerflow 를 활성화 시키기(TM210 – Working with Automation Studio).

신호가 지나는 길은 변수 모니터에서 **"Switch"** 변수의 값을 변경해서 볼 수 있다.

1) 온라인 연결을 하고 **"Lamptest"** 프로그램을 열기.

2) 모니터 모드를 활성화 시키기

3) **"Switch"**와 **"Lamp"** 변수를 변수 모니터에 추가하기

4) **"Switch"** 변수 설정

"Switch" 변수의 접속 상태가 만족하면, 신호는 "Lamp" 변수로 전달된다.

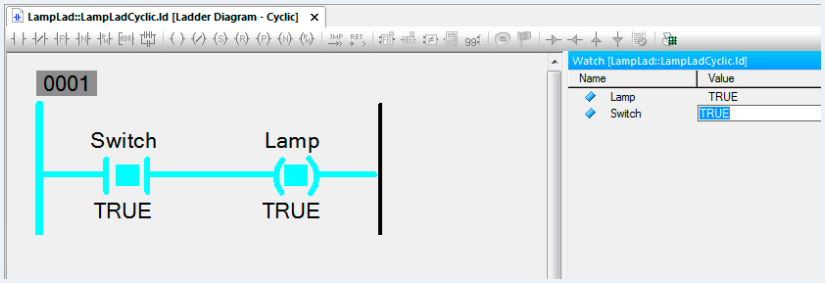

그림 48: Ladder Diagram 에서 활성화된 Power Flow

Diagnostics and service ₩ Diagnostics tools ₩ Monitors ₩ Programming languages in monitor mode ₩ Powerflow

5.2.3 Line coverage

Line coverage 가 텍스트 기반 프로그래밍 언어에서 활성화되면, 마커가 현재 실행중인 소스 코드 라인을 가리킨다.

이를 통해서 어떤 시간에 어떤 라인이 실행 중인지 정확하게 볼 수 있다. Line coverage 는 툴 바의 "Line Coverage" 아이콘을 선택하면 활성화된다.

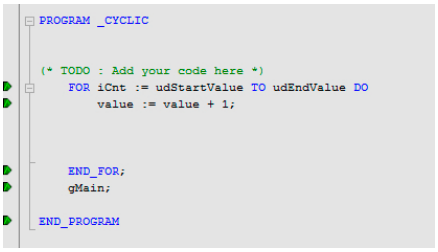

그림 49: Structured Text 에서 Line coverage

Diagnostics and service ₩ Diagnostics tools ₩ Monitors ₩ Programming languages in monitor mode ₩ Line coverage

5.2.4 IEC 체크 라이브러리

IEC 체크 라이브러리는 분할 작동, 범위 위반, 적절한 배열 접근, 메모리에 데이터 읽고 쓰기 등을 확인하는 다양한 함수들을 포함하고 있다.

각각이 실행되기 전에 알맞는 검사 함수는 프로그램(IEC 61131-3 언어 또는 Automation Basic 에 지원되는)에 의해 호출된다.

IEC 체크 라이브러리를 사용하면, 0 으로 나눠지거나 범위를 벗어난 에러 또는 잘못된 메모리 접근시 무엇을 어떻게 해야하는지 결정하기 위해 동적 변수(REFERENCE TO)를 사용할 수 있다.

Programming ₩ Libraries ₩ IEC Check library

5.2.5 소스 코드 디버깅

디버거는 프로그래머가 프로그램이나 라이브러리 소스 코드에서 에러를 쉽게 찾을 수 있도록 해준다.

Automation Studio 디버깅 가능성

- 변수를 모니터링 하는 동시에 한 줄씩 프로그램의 실행.
- 사용자가 결정한 브레이크 포인트(breakpoints)에서 어플리케이션을 정지.
- 호출된 함수로 들어가는 것. (예: 소스 코드가 동작가능한 상태에서 라이브러리 함수/ 함수 블록으로)

예제: 디버거를 사용해서 Structured Text 프로그램에서 에러 찾기

"dbgTest"라고 이름지어진 Structured Text 프로그램을 생성하라.

사이즈가 10 인 USINT 배열, "AlarmBuffer"을 추가하고 "dbgTest.var" 파일에 UINT 타입의 "index" 변수를 추가하라.

프로그램 사이클 파트에서, 어떤 값(예:10)으로 배열을 초기화시키는 루프를 사용하라.

아래의 프로그램 코드는 가장 일반적인 에러를 발생시킨다.

Program code

```
PROGRAM _CYCLIC
	FOR index := 0 TO 10 DO
		AlarmBuffer[index] := 10;
	END_FOR
END_PROGRAM
```

표 4: Faulty 프로그램 서브루틴

에러 설명: 배열이 10 개의 크기(0-9)로 선언되었는데 배열 크기가 초과되었다(0-10). 이런 에러의 종류는 처음 훑어 봤을 때는 찾기 힘들고 다음 변수 메모리 위치에 겹쳐 씌여지게 된다.

"dbgTest" 프로그램 생성하기

- Logical View 에서 새로운 Structured Text 프로그램 "dbgTest" 를 생성하라
- 그 프로그램의 변수 선언 다이얼로그 박스를 열고 변수을 생성하라.
  "AlarmBuffer" (데이터 타입 USINT[0.9]) 그리고 "index" (데이터 타입 UINT).
- 사이클 프로그램에 프로그램 코드를 입력해라.

프로그램이 시작되면, 배열의 모든 요소에 10 이 입력될 것이다.; 아래의 그림과 같이 프로그램이 작동할 것이다.

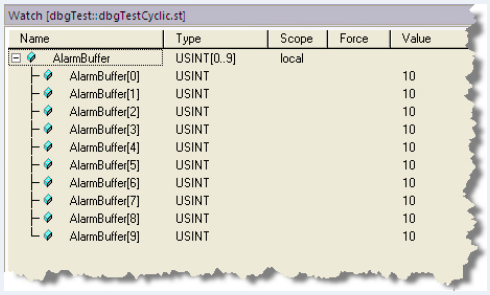

그림 50: 모니터 모드에서 Watch 창

변수 모니터에서 변수(그 변수의 값)뿐만 아니라 디버거에 있는 정보들을 가지고 에러 상황을 분석할 수 있다.

모니터 모드는 프로그램 에디터가 열리면 활성화 된다.

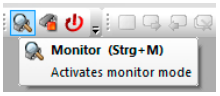

그림 51: 활성화된 모니터 모드

툴 바에서 디버거를 On/Off 할 수 있다.

그림 52: 디버거 on/off

Watch 창에 변수 추가하기

- 모니터 모드를 활성화 시킨 후, “AlarmBuffer” 변수를 Watch 창에 추가해라.

왼쪽 창에는 프로그램 코드가 보이며, 변수 모니터는 오른쪽에 보인다.

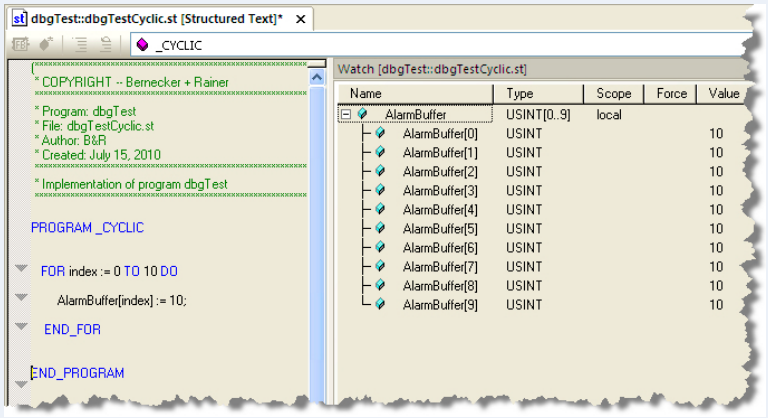

그림 53: 프로그램 에디터에서 모니터 모드

브레이크 포인트 (breakpoint) 설정

- FOR 루프의 첫 번째 줄로 커서를 이동시켜라.
- 메뉴의 <Debug> / <Toggle Breakpoint>를 사용하거나 <F9> 키를 눌러서 브레이크 포인트를 설정하라.

브레이크 포인트에 도달하면 타겟 시스템의 모든 어플리게이션 동작이 멈춘다.

Debugger 활성화시키기

- 디버거 활성화시키기: 디버거와브레이크 포인트는 메뉴에서 활성화시킨다.

디버거가 브레이크 포인트에 이르면, 그 줄에 노란 마커로 표시된다.

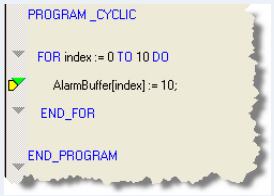

그림 54: 디비거에서 활성화된 줄

"Step into" 디버깅

- 변수 모니터에서 "AlarmBuffer" 배열의 값을 0 으로 바꿔라.
- "Step into" 명령어나 (<F11>) 키를 사용해서 프로그램 코드를 한 줄씩 실행할 수 있다. 실행중인 소스 코드 라인은 노란 마커로 표시된다.

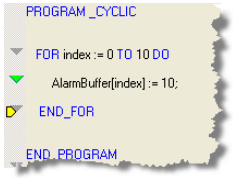

그림 55: 노란 마커로 표시된 실행중인 스텝(step)

<F11>키를 몇 번 누르면, 각각의 루프가 반복되어 배열의 값을 변화시킨다.

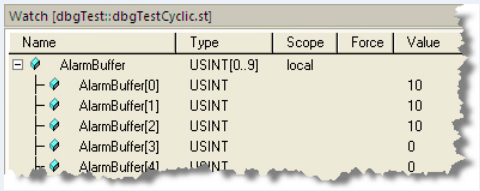

그림 56: 점진적으로 변수 입력하기

점진적인(Step-by-step) 실행

- 배열의 마지막 요소까지 새로운 값이 할당되도록 <F11>을 계속해서 클릭해라.

이 경우에, "index" 변수에 9 가 들어간다, 이때 배열([0..9]) 위쪽 범위 제한에 해당한다. 계속해서 <F11>을 누르면, 루프는 배열 범위 밖의 값을 한 번 더 반복할 것이다.

이 에러 종류는 IEC 체크 라이브러리에서 검출된다.

Diagnostics and service ₩ Diagnostics tool ₩ Debugger

5.2.6 상태 변수와 리턴 값 평가하기

함수에서 리턴된 값은 프로그램 내에서 값이 구해져야 한다.

리턴 값의 기능:
다음에 오는 예제는 함수 호출을 보여준다. 이 함수는 호출하는 동안 에러(ERROR)가 발생했는지 또는 프로그램 사이클(BUSY)에 접촉이 가능한지에 대한 상태를 리턴한다.

```
2:  (*Read information from an AR logger user module*)
    Logger.AsARLogGetInfo_0.enable := 1;
    Logger.AsARLogGetInfo_0.pName :=  ADR('usrlog');
    Logger.AsARLogGetInfo_0;  (*Call the Functionblock*)

    (*AsArLogGetInfo successful*)
    IF Logger.AsARLogGetInfo_0.status = 0 THEN
        Logger.Step := 0;
    ELSIF Logger.AsARLogGetInfo_0.status = ERR_FUB_BUSY THEN
        (*Busy*)
    ELSE  (*Go to Error Step*)
        Logger.Step := 10;
    END_IF
```

그림 57: 함수 블록의 상태 평가

함수의 상태(주황색 박스 안의 코드)가 정확하게 판별되지않는다면, 함수 또는 서브 시퀀스 프로그램이 제대로 실행되지 않는다.

5.3 프로그램에서 변수 사용하기

Logical View 에 있는 서로 다른 프로그램 내부에서 적절한 변수 사용여부는 cross-reference 리스트를 생성하거나 알려진 변수 이름을 정확하게 찾아내서 확인할 수 있다.

5.3.1 Cross reference

Cross-reference 리스트는 어떤 프로세스 변수, 함수, 펑션 블록이 프로젝트 어디에서 사용되는지 가리킨다.

Cross-reference 리스트는 선택할 수 있고 프로젝트가 컴파일(빌드)될 때 생성된다; 그 결과는 출력 창 “Cross Reference” 탭에서 보인다.

Cross-reference 리스트를 생성하기 위해서, 아래의 메뉴 옵션을 사용해서 활성화 시켜야 한다: <Project> / <Settings>. 이를 대신해서, Cross-reference 리스트는 메뉴 아이템 <Project> / <Build Cross Reference>를 사용해서 생성할 수도 있다.

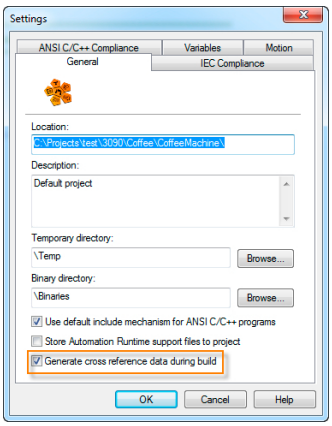

그림 58: 프로젝트 빌드 동안 cross-reference 목록 활성화

예제: cross reference 목록 생성

열려있는 프로젝트에 Cross-reference 리스트를 생성하라

1) 프로젝트 설정에서 cross-reference 리스트를 활성화.
2) 프로젝트 빌드.
3) 출력 창에서 cross-reference 리스트 체크.

출력 창에서 변수들과 그 특성에 대해 cross-reference 를 분석할 수 있다.

왼쪽 그림에서 변수를 선택하면, 접촉 타입와 사용처가 소스 코드 또는 오른쪽 그림의 I/O allocation 에 보여진다.

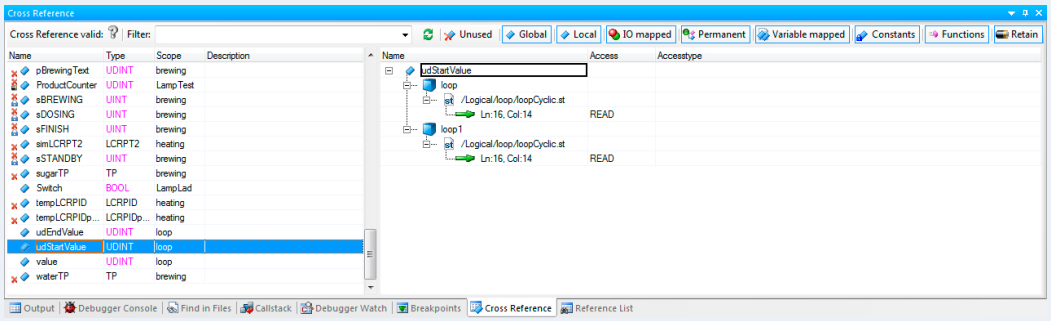

그림 59: cross reference 리스트

Project management ₩ The workspace ₩ Output window ₩ Cross reference
Project management ₩ The workspace ₩ Menus ₩ Project ₩ Build cross reference

5.3.2 파일 안에서 검색하기

이름, 제품 ID 또는 코멘트한 위치를 찾는다면, 프로젝트 파일에서 찾을 수 있다.

변수를 찾기위해서 메인매뉴에서 <Edit> / <Find and Replace – Find in Files> 를 누르거나, 단축키 <CTRL> + <Shift> + <F> 를 사용하라.

찾고자하는 단어를 다이얼로그 박스에 입력하면, 결과가 “Find in Files” 탭에 보일 것이다.

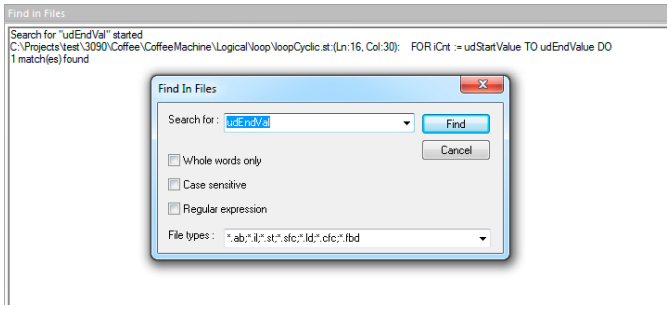

그림 60: 파일 찾기

출력 창에서 결과를 더블 클릭하면 각각의 소스 파일을 열수 있고 알맞는 위치에 커서가 놓인다.

6 서비스 준비

기계나 시스템의 어플리케이션 설정, 시운전, 테스트 그리고 추후에 진행되는 서비스를 진행 하는 동안 서비스 준비는 필요하다.

그림 61: SDM 시작 화면

6.1 System Diagnostics Manager (SDM)

System Diagnostics Manager (SDM)는 Automation Runtime V3.0 이상에서 지원되고 어느곳에서든(인트라넷 또는 인터넷) 인터넷 브라우저를 사용하여 컨트롤러를 진단할 수 있다.

진단을 위해 컨트롤러와 인터넷 연결만 필요하다.

Diagnostics and service ₩ Diagnostics tools ₩ System Diagnostics Manager

6.1.1 SDM 활성화 하기

SDM 은 새로운 프로젝트가 생성되면 자동으로 활성화된다. SDM 설정은 Physical View 에서 컨트롤러를 선택하고 마우스 우클릭 후 <Configuration>옵션을 사용하면 열 수 있다.

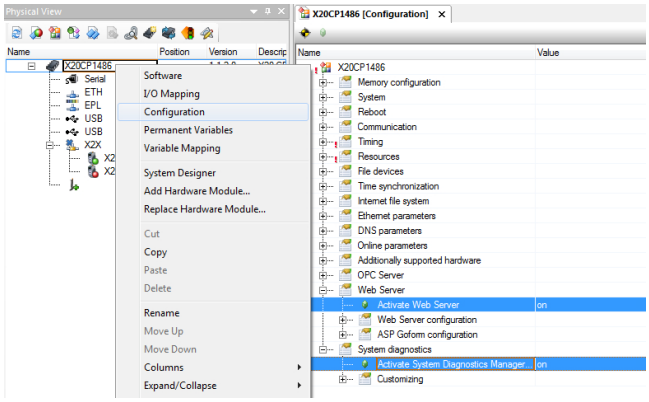

그림 62: 설정 열기, SDM 과 웹 서버 설정

System Diagnostics Manager 는 웹 서버 서비스를 요구하며, 이는 Automation Runtime 의 통합 컴포넌트이다.

예제: SDM 설정 체크

웹 서버와 System Diagnostics Manager 가 Automation Runtime (AR) 설정에서 사용가능한지 확인하라.

1) Physical View 에 있는 컨트롤러의 단축메뉴에서 AR 설정을 열기.
"Web Server" 그리고 "System Diagnostics" 옵션을 활성화 설정을 해야한다.

6.1.2 SDM 에 접속하기

System Diagnostics Manager 의 정보는 웹브라우저(web browser)를 사용하면 볼 수 있다.

접속하기 위해서, 타겟 시스템의 IP 주소를 알아야한다.

타겟 시스템 IP 주소는 컨트롤러의 Ethernet 설정에서 확인할 수 있다. 이미 컨트롤러와 연결이 되있다면, SDM 은 메뉴 옵션의 <Tools> / <System Diagnostics Manager> 메뉴를 선택해서 열 수 있다.

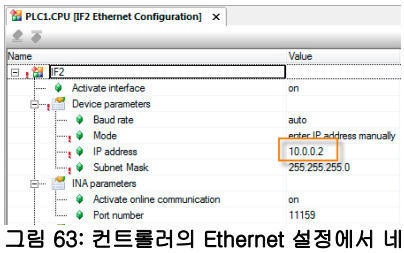

그림 63: 컨트롤러의 Ethernet 설정에서 네트워크 설정 확인하기

SVG (Scalable Vector Graphics) 서포트를 제공하기 위해서 플러그-인(plug-in)은 인터넷 익스플로러(버전 7.x 부터)를 통해 자동으로 설치된다.

Diagnostics and service ₩ Diagnostics tools ₩ System Diagnostics Manager ₩ FAQ ₩ SVG plug-in

예제: 웹 브라우저에서 SDM 에 접속하기

URL 을 입력하여 SDM 에 접속하라: 예 http://10.0.0.2/SDM

1) 웹 브라우저를 실행하라.

2) URL 을 입력하라.

URL 입력 후 <Enter> 를 누르면, SDM 페이지가 타겟 시스템에서 로드되고 브라우저에 보인다. SDM 왼쪽에 네비게이션이 있다.

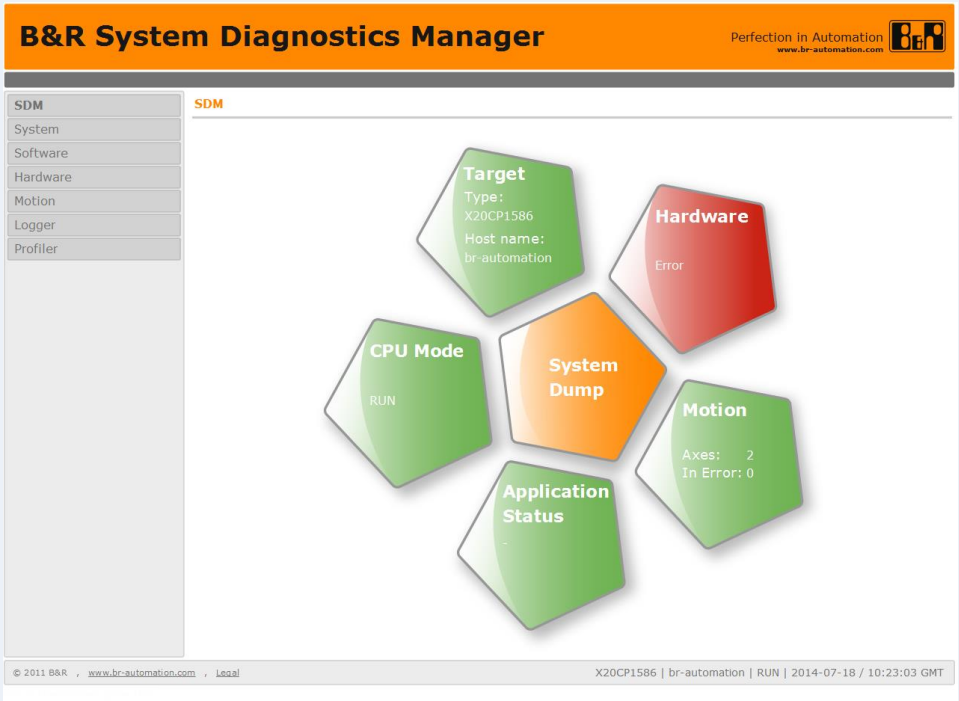

그림 64: 웹 브라우저에서 SDM 시작 페이지

예제: SDM 에서 Logger 항목을 저장하고 Automation Studio 에서 데이터를 분석하라.

상황: 시스템이 이유없이 서비스 모드에서 부팅되었다. 게다가 Automation Studio 는 사용할 수 없다. 이때 컨트롤러에서 SDM 이 활성화 되있다면, 컨트롤러와 TCP/IP 연결 후 웹브라우저를 통해서 System Diagnostics Manager 로 확인한다.

SDM 이 열리면, Logger 항목들은 검색 메뉴(Logger)에서 볼 수 있다. Logger 파일 **"$arlogsys"** 를 업로드하고 automation Studio 에서 Logger 파일을 열어라.

1) SDM 에 연결하고 "Logger" 페이지로 변경하라

2) "$arlogsys" 모듈을 업로드하라.

3) .br 확장자를 사용해서 파일을 저장하라.

4) Automation Studio 에서 Logger 을 열어라.

5) 확장명이 .br 인 파일을 열어라.

6) Logger 에 있는 에러 항목을 선택하고 <F1>키를 누르면 자세한 정보를 얻을 수 있다.

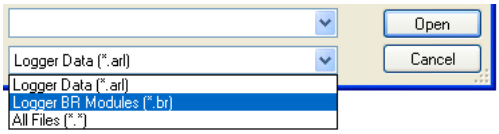

그림 65: .br 확장자 파일 타입

Automation Runtime 이벤트는 추가 정보 없이 SDM Logger 에 보인다. 이는 Automation Studio 에서 볼 수 있다.

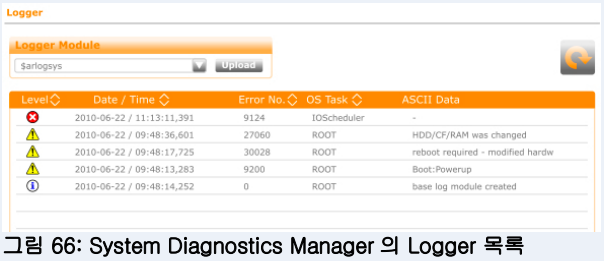

그림 66: System Diagnostics Manager 의 Logger 목록

온라인 데이터는 Automation Studio 의 Logger 에서 선택해제 되어야 한다; 반면에, 온라인 항목들 뿐만 아니라 .BR 모듈 안에 있는 항목들도 보일 것이다.

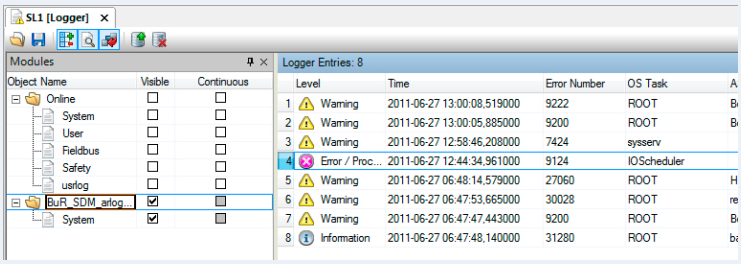

그림 67: 온라인 Logger 항목들 비활성화

6.2 배터리 상태 보기

어플리케이션에서 사용되고 있는 실시간 클락(clock)과 비휘발성 변수들 (retain, per-manent)을 위한 백업 배터리는 어플리케이션 자체에서 모니터링 할 수 있다.

그림 68: “4A0006.00-000” 리튬 배터리

배터리 교체 방법은 기계 / 시스템에 대한 서비스 설명서에 명시되어 있다. 배터리의 수명은 사용하는 컨트롤러 설명서에 명시되어 있다.

다양한 방법으로 배터리 수명을 알 수 있다:

- 어플리케이션의 AsHW 라이브러리 사용
- 컨트롤러의 I/O allocation 사용
- 온라인 정보 다이얼로그 사용
- System diagnostics manager 사용

I/O allocation 은 Physical View 에서 컨트롤러 단축메뉴 <I/O Allo-cation> 을 누르면 열린다. “BatteryStatusCPU”에 있는 변수는 필요에 따라 어플리케이션에서 쓰여질 수 있다.

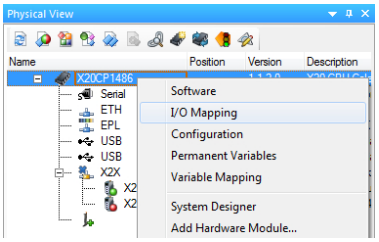

그림 69: 컨트롤러의 I/O allocation

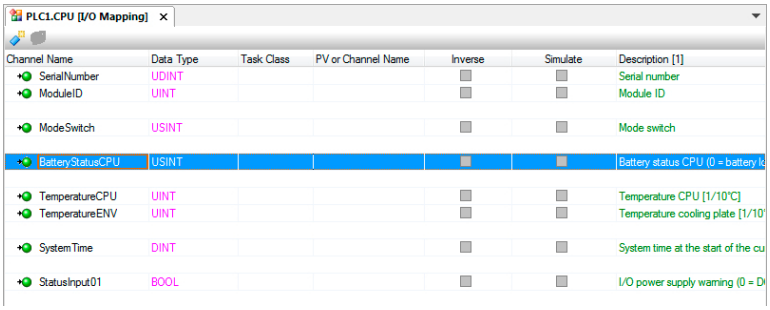

그림 70: 컨트롤러 I/O allocation 에서 배터리 상태 보기

컨트롤러 I/O mapping 에 있는 데이터 포인트 상태 값은 모니터 모드에서 볼 수 있다.

4.3 “I/O 모니터링과 force” 를 참고하라

6.3 Runtime Utility Center 진단 툴

런타임 유틸리티 센터(Runtime Utility Center) 프로그램은 Automation Studio 에서 CompactFlash 로 실행할 수 있는 어플리케이션을 복사하는데만 사용되지않는다.

런타임 유틸리티 센터는 시운전, 유지, 진단, 서비스 등에 전 과정에서 유용하게 사용된다.

런타임 유틸리티 센터는 **<Tools> / <Runtime Utility Center>**를 선택하면 Automation Studio 에서 시작할 수 있다.

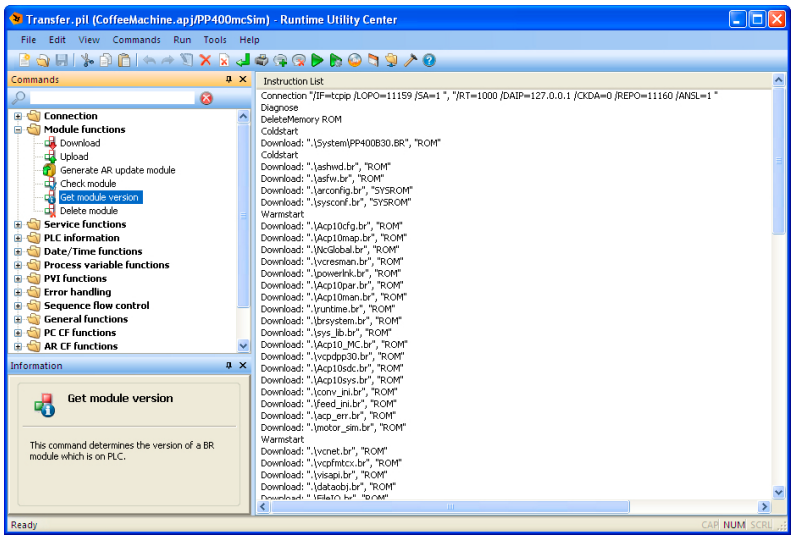

그림 71: 런타임 유틸리티 센터

Automation Studio 프로젝트를 빌드한 후, 런타임 유틸리티 센터를 열면 프로젝트 리스트 파일 (.pil)이 생성된다.

Diagnostics and service ₩ Service tools ₩ Runtime Utility Center

6.3.1 변수 값 백업하고 복구하기

런타임 유틸리티 센터의 기능 중에는 컨트롤러로부터 변수 값을 로드하고 이것들을 다시 불러올 수 있다.

예제: 변수 값 저장하기.

시스템에 물리적 손상이 가해졌을 때, CPU 를 교체해야 한다. 예를 들어 레시피 변수(recipe variables) 또는 다른 프로세스 데이터가 삭제되는 것을 막기 위해서, CPU 에 필요한 정보는 런타임 유틸리티 센터를 사용해서 업로드하고 나중에 새로운 CPU 에 옮긴다.

“Loop” 테스크에서 변수 값을 저장할 수 있는 런타임 유틸리티 센터 리스트를 생성하라.

1) Automation Studio 에서 런타임 유틸리티 센터를 실행하라.

2) 메인 메뉴에서 <File> / <New>를 눌러서 새로운 리스트를 생성하라.

3) <Command> / <Connection> 을 선택해서 연결을 위한 커멘드를 입력하라.
타겟 시스템의 IP 주소는 연결 설정에 입력되어야 한다.

4) <command> / <Process variable function> / <Variable list>를 선택해서 변수 리스트를 로딩하는 커멘드를 입력하라

5) <F5>를 눌러서 실행하라.

Diagnostics and service ₩ Service tools ₩ Runtime Utility Center ₩ Operation ₩ Commands ₩ List functions

Diagnostics and service ₩ Service tools ₩ Runtime Utility Center ₩ Operation ₩ Commands ₩ Establish connerction, wait for reconnection

Diagnostics and service ₩ Service tools ₩ Runtime Utility Center ₩ Operation ₩ Menus ₩ Start

"Loop" 테스크의 변수들은 명시된 디렉토리와 파일 이름을 사용해서 변수 값이 백업된다.

예제: 변수 값 불러오기

마지막 테스크에 백업 된 변수 값은 런타임 유틸리티 센터를 사용해서 컨트롤러에 옮겨야한다.

변수 값을 불러오는 런타임 유틸리티 센터를 생성하라.

1) Automation Studio 에서 런타임 유틸리티 센터를 열고 메뉴의 <File> / < New>를 눌러서 새로운 리스트를 생성하라.

2) <Commands> / <Connection>을 눌러서 연결하는 커멘드를 입력하라.

3) 연결 설정에 타겟 시스템의 IP 주소를 써야한다.

4) <Command> / <Process variable functions> / <Transfer variable list to PLC>를 눌러서 변수 리스트를 쓰기 위한 커멘드를 입력하라.
마지막 테스크에 저장된 변수 리스트는<Browse> 다이얼로그 박스에서 선택할 수 있다.

5) <F5> 를 눌러서 실행하라.

파일에 저장된 변수 값은 "Loop"에 알맞는 변수에 쓰여진다.

모든 테스크의 모든 변수가 백업되고 컨트롤러에 전달되면, 계속해서 변수에 덮어 쓰여져서 프로세스가 예상치 못한 행동을 일으킬 수 있다.

이런 상황이 필요하다면, 처음에는 서비스 모드에서 컨트롤러를 작동시키는 것을 권장한다.

6.3.2 런타임 유틸리티 센터를 사용해서 프로젝트 전달하기

완성된 런타임 유틸리티 센터 프로젝트 목록은 서비스와 기계 설치를 위해서 실행가능한 어플리케이션으로 옮겨질 수 있다.

이를 위해 PC 와 온라인 연결이 되어 있는 CPU 가 있어야 한다.

예제: 컨트롤러 재설치

컨트롤러는 Automation Studio 를 사용하지 않고도 재설치 할 수 있다. 프로젝트가 빌드된 후에 Automation Studio 에서 런타임 유틸리티 센터를 시작하라. 설치 패키지는 런타임 유틸리티 센터에서 <Tools> / <Generate installation package>를 누르면 생성가능하다.

요구사항:

- 컨트롤러와 실행가능한 Autoamation Runtime 버전을 포함하는 CompactFlash 를 갖춰야한다.
- CPU 의 IP 주소는 알고 있어야 한다.

설치 패키지 생성하기

- Automation Studio 에서 프로젝트를 컴파일해라.
- <Tools> / <Runtime Utility Center>를 선택해서 Automation Studio 에서 런타임 유틸리티 센터를 시작하라.
- < Tools > / <Generate installation package>를 선택해서 런타임 유틸리티 센터 설치 이미지를 생성해라.
- 목적 디렉토리가 지정된 후, <Start>를 누르면 생성 프로세스가 시작된다.
  생성 프로세스가 끝나면 런타임 유틸리티 센터가 닫힌다.

프로젝트 설치를 위해 실행가능한 이미지는 Automation Studio 의 도움 없이 생성된다.

설치 패키지 전달하고 실행하기

- 윈도우 익스플로러를 열고 주소창에 설치 패키지가 생성된 주소를 입력한다.
- “Start.bat” batch 파일을 실행시켜라.

런타임 유틸리티 센터 설치를 다른곳에 옮기기 위해, CD 생성 프로세스 동안 지정된 주소의 내용은 CD 에 구워지거나 flash drive 에 복사될 수 있다.

“Start.bat” 파일은 새로운 시스템이 PC 에서 작동되는데 필요하다.

Diagnostics and service ₩ Service tools ₩ Runtime Utility Center ₩ Creating a list / data medium

7 요약

문제와 에러를 찾아내는 다양한 진단 툴이 있다.

분석적으로 생각해서 적합한 진단 툴을 사용해야 한다.

그림 72: 진단(Diagnostics)

이런 진단 툴을 효율적으로 사용하기 위해서는, 상황에 대한 전반적인 이해, 일반적인 상태를 명확히 설명하고 상황을 확인합니다.

그러면 명확하고 자세하게 분석할 수 있다. 잠재적 에러에 대한 포괄적인 이해는 에러 소스에 대한 많은 가능성을 감소시킬 수 있고 이는 계속해서 발생하는 어떤 에러도 상당히 쉽게 바로잡을 수 있다.

8 부록

8.1 “Loop” 샘플 프로그램

“Loop” 프로그램

“Loop” 프로그램은 Structured Text 프로그래밍 언어로 만들었다. FOR 루프의 시작과 끝을 변수로 설정하여 값을 변경할 수 있다. 끝 값을 증가시킴으로써 CPU 로드를 증가시켜서 cycle time violation 을 발생 시킬 수 있다.

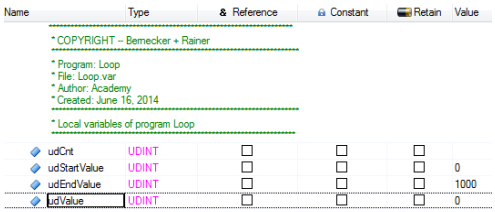

그림 73: “Loop” 변수 선언

```
PROGRAM _INIT
    udEndValue := 50;
END_PROGRAM

PROGRAM _CYCLIC

    (* implementation of program Loop *)
    FOR udCnt := udStartValue TO udEndValue DO
        udValue := udValue + 1;
    END_FOR

    gMain := gMain + 1;

END_PROGRAM
```

그림 74: Structured Text 로 “Loop” 실행

Training modules
TM210 – Working with Automation Studio
TM213 – Automation Runtime
TM223 – Automation Studio Diagnostics
TM230 – Structured Software Development
TM240 – Ladder Diagram (LD)
TM241 – Function Block Diagram (FBD)
TM242 – Sequential Function Chart (SFC)
TM246 – Structured Text (ST)
TM250 – Memory Management and Data Storage
TM400 – Introduction to Motion Control
TM410 – Working with Integrated Motion Control
TM440 – Motion Control: Basic Functions
TM441 – Motion Control: Multi-axis Functions
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Initial Commissioning of Motors
TM500 – Introduction to Integrated Safety
TM510 – Working with SafeDESIGNER
TM540 – Integrated Safe Motion Control
TM600 – Introduction to Visualization
TM610 – Working with Integrated Visualization
TM630 – Visualization Programming Guide
TM640 – Alarm System, Trends and Diagnostics
TM670 – Advanced Visual Components
TM800 – APROL System Concept
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM890 – The Basics of LINUX
TM920 – Diagnostics and service
TM923 – Diagnostics and Service with Automation Studio
TM950 – POWERLINK Configuration and Diagnostics

TM1010 – B&R CNC System (ARNC0)
TM1110 – Integrated Motion Control (Axis Groups)
TM1111 – Integrated Motion Control (Path Controlled Movements)
TM261 – Closed Loop Control with LOOPCONR
TM280 – Condition Monitoring for Vibration Measurement
TM480 – The Basics of Hydraulics
TM481 – Valve-based Hydraulic Drives
TM482 – Hydraulic Servo Pump Drives